

# REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-00-

ources.  
ollection  
Reports  
shall be

The public reporting burden for this collection of information is estimated to average 1 hour per response, including gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that any subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OI PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 12-05-2000		2. REPORT TYPE Final Technical Report		3. DATES COVERED (From - To) 01-02-95 to 31-05-98	
4. TITLE AND SUBTITLE Shop Floor Scheduling for Program Depot Maintenance Considering Stochastic Repair Needs				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER F49620-95-1-0172	
				5c. PROGRAM ELEMENT NUMBER	
				5d. PROJECT NUMBER	
6. AUTHOR(S)  Gemmill, Douglas, D.				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Iowa State University 2019 Black Engineering Ames, IA 50011				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NM 110 Duncan Ave. Suite B115 Bolling AFB, DC 20332-0001				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>DISTRIBUTION STATEMENT A</b> Approved for Public Release Distribution Unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The research supported by this grant investigated new methods to schedule aircraft depot maintenance activities while considering resource constraints and the stochastic nature of task durations. Depot maintenance can be characterized as a large and very complex project scheduling problem. The research resulted in the development of a new method to identify the critical path within resource-constrained project scheduling problems, heuristic search methods that improved overall project completion time by an average of nine percent on test problems, look ahead scheduling methods which further improved project duration, heuristics for improved scheduling when using common resources on multiple projects simultaneously, and heuristics which improved the scheduling of resource-constrained problems when multiple execution modes are available for each task. A windows-based software application was developed which allows a practitioner to use a combination of these methods in order to create good solutions to this complex depot maintenance scheduling problem.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include area code)

20000925 090

# **FINAL TECHNICAL REPORT**

**Submitted to the Air Force Office of Scientific Research**

## **Shop Floor Scheduling for Program Depot Maintenance Considering Stochastic Repair Needs**

Grant Number F49620-95-1-0172

submitted by

Douglas D. Gemmill  
Associate Professor  
Department of Industrial & Manufacturing Systems Engineering  
Iowa State University

20000925 090

## TABLE OF CONTENTS

<b>Abstract</b>	<b>1</b>
<b>Executive Summary</b>	<b>1</b>
<b>Identifying the Critical Path in Resource Constrained Projects</b>	<b>5</b>
<b>Using a Simulated Annealing Algorithm to Schedule Activities of Resource-Constrained Projects</b>	<b>12</b>
<b>Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects</b>	<b>20</b>
<b>Improving Resource-Constrained Project Schedules with Look Ahead Techniques</b>	<b>30</b>
<b>Multiple Resources Constrained Multiple Projects</b>	<b>39</b>
<b>Optimizing the Project Duration of Multiple Mode Resource-Constrained Projects using Simulated Annealing</b>	<b>47</b>
<b>Scheduling Software Developed</b>	<b>58</b>
<b>Aggregate of References for All Papers</b>	<b>63</b>

# **Shop Floor Scheduling for Program Depot Maintenance Considering Stochastic Repair Needs**

## **Abstract**

The research supported by this grant investigated new methods to schedule aircraft depot maintenance activities while considering resource constraints and the stochastic nature of task durations. Depot maintenance can be characterized as a large and very complex project scheduling problem. The research resulted in the development of a new method to identify the critical path within resource-constrained project scheduling problems, heuristic search methods that improved overall project completion time by an average of nine percent on test problems, look ahead scheduling methods which further improved project duration, heuristics for improved scheduling when using common resources on multiple projects simultaneously, and heuristics which improved the scheduling of resource-constrained problems when multiple execution modes are available for each task. A windows-based software application was developed which allows a practitioner to use a combination of the methods in order to create good solutions to the complex scheduling problem.

## **Executive Summary**

### **General background**

One location at which the United States Air Force (USAF) performs programmed depot maintenance (PDM) of aircraft is Warner Robins Air Logistics Center (WR-ALC). The methods proposed for sequencing activities of resource-constrained project schedules were developed as part of a study of PDM at WR-ALC. Specifically, the maintenance of large-body cargo aircraft was examined. As part of the study, the methods used by three major airlines for their depot maintenance activities were examined for comparison. Depot maintenance involves the disassembly and inspection of almost all aircraft systems. Literally thousands of activities are performed, both routine and those resulting from discrepancies found during inspections. This process can be represented as a very large resource-constrained project schedule. The problem is to find the best sequence in which to complete these activities such that the time on station for each aircraft is reduced.

Commercial airlines and the USAF use similar methods to sequence the project activities. Each uses a system in which every activity associated with the maintenance project is referenced on a card. The commercial airlines visited chose to sort and display the work cards on a large slot board on or near the production floor; whereas, WR-ALC placed them in high boys, or work benches. Although both the commercial airlines and WR-ALC employ a card system, each monitors the progress of maintenance projects differently. In some cases an attempt is made to use a software program to assist with management of the project; however, those actually conducting the maintenance operations infrequently use these tools.

Some of the tools have the capability to indicate critical activities, but none appear to have the capability to identify an actual schedule, or sequence, of activities. Instead, the sequence in which to perform the maintenance tasks is determined based on which activities those in charge of maintenance judge to be important. That is, much of the schedule is determined based solely on the experience of the shop floor supervisors or crew chiefs. It is common for supervisors and mechanics to choose the order in which activities are completed without concern for critical activities. This "cherry picking" method of scheduling leads to a variety of problems. One such problem is the possibility of delaying critical activities, which in turn could delay the entire maintenance project.

It should be noted that some project scheduling packages have the capability to determine the total utilization of required resources at any point in time of a project for which the sequence and timing of activities has been set. A manager can use this capability to determine when he is under or over utilizing any of his resources. Required adjustments can then be made to the schedule in an attempt to level the utilization of resources. The important difference between this and the method we propose is that our method automatically creates a schedule while taking resource constraints into consideration. However, once we have created the schedule, the manager could again look at resource utilization and make further adjustments. In addition, the methodologies and software we developed allow for the inclusion of randomness in the project.

#### **Theses resulting from the research**

Tsai, Ying-Wei. *Resource constrained project scheduling using simulated annealing and tabu search*. MS Thesis, Department of Industrial and Manufacturing Systems Engineering, Iowa State University (1996).

Edwards, Michelle L. *Improving constraint-based scheduling with look ahead techniques*. MS Thesis, Department of Industrial and Manufacturing Systems Engineering, Iowa State University (1997).

Kim, Seong-Ryong. *Multiple resources constrained multiple projects*. MS Thesis, Department of Industrial and Manufacturing Systems Engineering, Iowa State University (1998).

Rangaswamy, Ravikumar. *Optimizing the project duration of multiple mode resource-constrained projects using simulated annealing*. MS Thesis, Department of Industrial and Manufacturing Systems Engineering, Iowa State University (2000).

#### **Publications resulting from the research**

Y. W. Tsai and D. D. Gemmill, Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects, *European Journal of Operation Research*, **111** (1) (1998) 131-141.

Gemmill, D.D. and Tsai, Y. W., Using a Simulated Annealing Algorithm to Schedule Activities of Resource-Constrained Projects, *Project Management Journal*, **28** (4) (1998) 8-20.

Gemmill, D.D. and M.L. Edwards, Improving Resource-Constrained Project Schedules with Look-Ahead Techniques, *Project Management Journal*, **30** (3) (1999) 44-55.

Rangaswamy, R. and D.D. Gemmill, Optimizing the project duration of multiple mode resource-constrained projects using simulated annealing. Submitted to *Project Management Journal*.

### **Most significant advances from the research**

New method to identify the critical path for resource-constrained project scheduling problems (usually only technology constraints are considered). Allows for quick determination of resource-constrained critical path. This promotes the use of iterative optimization routines to search for optimal solutions due to the increased speed with which each iteration can be run.

Developed heuristic applications of simulated annealing and tabu search to find good solutions to resource-constrained project scheduling problems. SA showed a 9% average improvement in project duration over use of existing heuristics. Tabu also showed about a 9% improvement over existing heuristics. Both SA and tabu applications found optimal solutions in over 90% of the test problems.

Developed "look ahead" techniques to further improve on the solutions found using the existing or newly developed heuristics. Potential to further reduce project completion time by as much as 8%.

Problem extended to case where common resources are required to be shared among several projects simultaneously. Developed a heuristic and an application of SA to search for the best solution.

Developed three new heuristics used to search for good solutions to the resource-constrained problem when multiple execution modes for each task are available.

### **Faculty supported by the research**

Douglas D. Gemmill  
Associate Professor  
Department of Industrial & Manufacturing Systems Engineering  
Iowa State University

**Graduate students supported by the research**

Ying-Wei Tsai, M.S.  
Michelle Edwards, M.S.  
Seong-Ryong Kim, M.S.  
Ravikumar Rangaswamy, M.S.

Suttira Thanyavanich  
Wanida Kananam  
Yi-Chiuan Lai  
Lori Melaas

**Undergraduate students supported by the research**

Matthew Saxton  
Dan Toft  
Bill Leuenberger

# **Identifying the Critical Path in Resource Constrained Projects**

by Ying-Wei Tsai and Douglas D. Gemmill

Available as Working Paper #96-131

Industrial & Manufacturing Systems Engineering  
2019 Black Engineering  
Iowa State University  
Ames, Iowa 50011  
Tele: 515-294-8731  
n2ddg@iastate.edu

*The following is a condensed version of the above paper.*

## **Introduction**

Over the years, several techniques have been developed for the purpose of planning, scheduling, and controlling projects. The critical path method (CPM) and the program evaluation and review technique (PERT) are the most widely used techniques. Using these techniques a critical path is determined which does not consider the resource requirements of a project. A striking shortcoming of CPM/PERT is the inability of dealing with a scheduling problem when the required resources are in limited supply. Only a few researchers have considered the criticality of an activity in a resource constrained project.

In a traditional network model of a project, the activities are connected by links which correspond to technological relationships. The critical path in the traditional CPM/PERT technique is therefore defined as the path connected by technological relationships with the largest expected duration. This definition is no longer valid when the available resources are limited. The critical path found under the assumption of unlimited resources is not necessarily the critical path in a resource constrained project. The most damaging error is that activities initially found to have free slack time when resource constraints are not considered, become critical activities when constraints are included. To find the critical path in a resource constrained project, both technological relationships and resource constraints must be considered. The critical path in a resource constrained project is defined as the path connected by both technological relationships and shared resource relationships with the largest expected duration.

Woodworth and Shanahan [1988] claimed that they can correctly calculate slack times of activities with their proposed procedures. During a forward pass through the CPM network, each activity is given a resource sequence label which records the resource being used and the order of its use. In addition to the resource labels, a new linkage mechanism is used to examine the history of resource utilization at each resource allocation during the backward pass to calculate the latest start time. The new linkage mechanism frequently requires the creation of dummy activities. Continuing to examine the history of the resource

utilization and adopting the new linkage mechanism during the backward pass causes unnecessary extra burden.

Bowers [1995] presented a revised method of calculating the slack times of activities of a resource constrained project. Some links called resource links are explicitly created to trace the use of resources, noting the resource dependencies and hence identifying the critical path. A drawback of the revised method proposed by Bowers is that the resource requirements must be checked and validated during the backward pass. This requires the creation and maintenance of a resource utilization database and complicates the backward pass, especially when there are a large number of activities. The motivation for the creation of the proposed precedence links is to significantly simplify the backward pass. The precedence links defined in this paper ensure that the order in which activities are allocated resources is the same in both the forward pass and in the backward pass, and the creation and maintenance of a resource utilization database is not required.

## Methodology

Bowers' revised method for identifying the critical path of a resource constrained project includes six steps:

- Step 1.** Forward pass, ignoring resources. Calculate the earliest start time and the earliest completion time.
- Step 2.** Backward pass, ignoring resources. Calculate the latest start time and the latest completion time.
- Step 3.** Forward pass, with resources. Use a minimum latest start rule to assign resources to activities. Calculate the earliest start time and the earliest completion time and note resource utilization history.
- Step 4.** Add resource links.
- Step 5.** Backward pass, with resources, including resource links. Calculate the latest start time and the latest completion time.
- Step 6.** Compare earliest start time and latest start times.

The same calculations, as used in basic CPM/PERT, are performed in **Step 1** and **Step 2**. In **Step 3**, the minimum latest start (MINLS) heuristic for resource constrained projects is used to assign resources to activities. Then the earliest start time (EST) and the earliest completion time (ECT) can be determined. In addition, the history of resource allocation for each activity is noted in **Step 3** and this information is then used to add resource links in **Step 4**. The resource links explicitly represent the resource dependencies in the project network. In **Step 5** a backward pass incorporating the resource links provide the latest start time (LST) and the latest completion time (LCT) with the resource constraints. The critical activities and then the critical path are determined in **Step 6** by comparing the EST and LST.

Assume that many of the activities require the use of the same resources. A major drawback of the revised method proposed by Bowers is that the resource requirements must be checked and validated during the backward pass. To check the resource requirements and validate the resource availability, a resource utilization database has to be maintained while

scheduling. When resources are required to complete an activity, two operations must be completed. First, query the resource utilization database for the amount of resources which are not in use. Second, if the required resources are available, change the status of the resources to "in use." After an activity has been completed, the acquired resources must be released. The above process requires maintenance of a resources utilization database, which will cause heavy burden when many kinds of resources are required or when a project contains a large number of activities. With the addition of the proposed precedence links, the resource requirements no longer need to be considered during the backward pass.

Even though with Bowers' method we have now added the resource links to the project network, resource requirements still need to be considered during the backward pass. The addition of precedence links will eliminate the need to consider resource requirements during the backward pass. The procedures for creating the resource links and precedence links during the forward pass are presented in the following paragraphs. Note that Bowers used the minimum latest start (MINLS) heuristic to assign resources to activities for resource constrained projects. We will use the minimum slack first (MINSLK) heuristic in our illustration of how to add resource links and precedence links. Therefore, we are using a small modification of Bowers' method. However, the heuristic used to assign resources (MINLS or MINSLK) has no effect on Bowers' use of resource links to identify the critical path.

Assume that the project network has only one starting node denoted as S, and only one ending node denoted as T. Initially, no nodes are marked with start times. The fact that the start time for a node is unknown is denoted by saying that it is in set U. Two attributes are maintained for each resource, an attribute *AssignedTo* whose value is the number of the node to which this resource is assigned and an attribute *ReadyTime* which is the time when the resource will be available. The initial value for *AssignedTo* is the node S and the initial value of *ReadyTime* is zero.

Initially, the start time of node S is set to zero and node S is moved from set U to a list W. Nodes in the list W represent activities for which all predecessors have been scheduled, waiting for resources to be available. The nodes in list W are ordered according to the priorities calculated by a specified rule. For example, the priority calculated by the MINSLK rule is that a smaller value of slack time has a higher priority. Once resources are allocated to an activity in list W based upon its priority, the activity will be moved from list W to list C. The nodes in list C are ordered according to the activity completion times. When a node becomes a member of set C, this means that the activity has been completed. The EST and the ECT are calculated and resource links and the precedence links are created. After the forward pass has been completed, the resource links and precedence links are added to the project network. During the backward pass, the LST and the LCT are calculated using exactly the same procedure as that of CPM/PERT without the need to consider resource requirements. The algorithm for determining resource links and precedence links is as follows:

## **The algorithm for determining the resource links and precedence links.**

---

Place all nodes in set U.

Set the attribute AssignedTo of all resources to be node S.

Set the attribute ReadyTime of all resources equal to 0.

Set the initial value of variable ResReadyTime to be 0.

Move S from set U to list W.

Let the initial value of C\_ACT, the completed activity, be S.

While not all activities have been scheduled do

    While the resources are available for the first node, denoted as NEXT, in list W do

        Set the EST of NEXT to be ResReadyTime.

        Compute ECT of NEXT.

        Move NEXT from list W to list C.

        If C\_ACT is not equal to S and NEXT is not equal to T then

            Add a precedence link from C\_ACT to NEXT.

        endif

        Allocate resources to NEXT.

        For each resource allocated to NEXT do

            Store the attribute AssignedTo in variable FROM.

            Set the attribute AssignedTo to be NEXT.

            if FROM is not equal to S then

                Add a resource link from FROM to NEXT

            endif

            Set the attribute ReadyTime to be the ECT of NEXT.

        done

    done

    Remove the first node of list C, store in the variable C\_ACT.

    Release the resources acquired by C\_ACT.

    Let ResReadyTime be the ECT of the node C\_ACT.

    Find the nodes whose all successors had been completed, move these nodes from set U to list W.

done

---

## Results

The 110 test projects assembled by Patterson [1984] were used to validate the proposed algorithm. These 110 projects represent an accumulation of multiple resource constrained problems existing in the literature. The number of resources required per activity vary between one and three. Of these 110 projects, 103 projects require 3 different types of resources, 3 projects require 2 different types of resources, and 4 projects require only one type of resource. The number of activities per project varies between 5 and 49. The proposed algorithm correctly identifies the resource-constrained critical path for each of the 110 project networks provided by Patterson.

Two comparisons were made to illustrate the relation between the number of precedence links created, the number of different resource types required and the number of activities in a project. Each of the 110 projects were scheduled using the MINSLK heuristic and activity durations were assumed to be deterministic. A comparison of number of resource links and precedence links required for each of the 110 projects based on the number of different resource types required is given in the table below. The ratio of the average number of precedence links required to the sum of the average number of arcs in the original project plus the average number of resource links required significantly decreases as the number of different resource types required increases. This indicates that in projects which need a large number of different types of resources the precedence links become a smaller part of the total links needed during the backward pass.

**A comparison of the number of resource links and precedence links required for Patterson's 110 projects based on the number of resource types required.**

Resource type(s) required	a. Average number of technological links	b. Average number of resource links required	c. Average number of precedence links required	$\frac{c}{(a+b)} \times 100\%$
1	20.25	20.75	13.50	32.93%
2	24.00	36.00	13.33	22.22%
3	36.24	105.58	22.61	16.48%

The table below shows a comparison of the number of resource links and precedence links required for Patterson's 110 projects based on the number of activities. The ratio is lower for projects which have more activities.

**A comparison of the number of resource links and precedence links required for Patterson's 110 projects based on the number of activities.**

Number of activities	a. Average number of technological links	b. Average number of resource links required	c. Average number of precedence links required	$\frac{c}{(a+b)} \times 100\%$
5,6,7	6.00	7.40	4.80	36.09%
11,12,16	15.33	18.33	10.33	33.88%
20,21,25	32.87	94.76	20.43	16.10%
33,49	71.00	202.10	44.08	17.10%

**Application to C-141 Aircraft Depot Maintenance**

Warner Robins Air Logistics Center (WR-ALC), located at Robins Air Force Base, Georgia, performs depot maintenance on C-141 Aircraft. There are three basic standard project schedules that are utilized depending upon the type of depot maintenance to be performed on a particular aircraft. Each standard package is modified for a given aircraft depending upon the types of discrepancies discovered during initial inspection of the airframe, engines, avionics equipment, etc. Presently WR-ALC utilizes a software package called Programmed Depot Maintenance Scheduling System (PDMSS) to assist in scheduling the activities required to complete depot maintenance. However, when determining the critical path and developing a schedule for the maintenance activities, PDMSS is unable to consider the effects of resource constraints. Instead, PDMSS provides a GANTT Chart that assumes unlimited resource availability. Therefore, supervisors and crew chiefs must determine the actual sequence in which to complete the activities without real knowledge of which activities fall on the critical path. With our proposed algorithm, the critical activities can be determined.

The three basic standard project schedules utilized by WR-ALC for C-141 Aircraft consist of 107 to 168 activities with four different basic types of resources required to complete the network. Most of the activities on these networks are actually an aggregate of several related activities. Activities may require only one type of resource or may require all four types for completion. The number of each type of resource required for each activity varies between one and five. The number of technological links, resource links, and precedence links required for the three standard networks is shown in the table below. When using our algorithm to determine the critical path about 17 to 20% additional links are required. This is similar to our previous results in that as the overall complexity of the network increases, the proportional number of precedence links required decreases.

**A comparison of the number of resource links and precedence links required for Warner Robins Air Logistic Center's C-141 Projects.**

Project	a. Number of technological links	b. Number of resource links required	c. Number of precedence links requir	$\frac{c}{(a+b)} \times 100\%$
1	215	325	106	19.63%
2	406	538	158	16.74%
3	420	529	164	17.29%

**Conclusion**

Bowers proposed a revised method which can calculate the criticality of an activity in a resource constrained project by adding resource links and considering the resource requirements during the backward pass. For a project in which resources are highly utilized and many activities require the same resources, the maintenance of a resource database can be a complicated and time consuming job. An additional link called a precedence link was proposed to eliminate the need to maintain a resource database. This greatly reduces the complexity of the backward pass, allowing the standard CPM/PERT method to be used.

The proposed algorithm correctly identifies the critical path for the 110 project networks provided by Patterson. The additional precedence links make up only a small part of the total number of links that need to be considered during the backward pass. In some cases, each of the precedence links are identical to some of the original network or resource links. The algorithm has also been applied to three resource constrained project networks provided by Warner Robins Air Logistics Center.

It is important to note that the proposed algorithm provides us with a tool that can now be used with other methods in an attempt to find the optimal sequence for scheduling the activities of resource constrained project networks. Most search algorithms, such as simulated annealing or Tabu search, require a great number iterations in order to approach finding the optimal solution to a problem. If the goal is to find the sequence that minimizes the total project length, then an efficient method to determine the critical path is necessary. The proposed method provides us with an efficient (simple and fast) method to determine the critical path.

# **Using a Simulated Annealing Algorithm to Schedule Activities of Resource-Constrained Projects**

by Douglas D. Gemmill and Ying-Wei Tsai

Published in *Project Management Journal*, 28 (4) (1997) 8-20.

Industrial & Manufacturing Systems Engineering  
2019 Black Engineering Annex  
Iowa State University  
Ames, Iowa 50011  
Tele: 515-294-8731  
n2ddg@iastate.edu

*The following is a condensed version of the above paper.*

## **Introduction**

CPM and PERT are well known project planning techniques used to determine the critical path of a project and hence the total time required to complete the project. These methods can be used to efficiently schedule the sequence of activities that make up a project assuming that there are no resource constraints. However, the ability of PERT/CPM to deal with variations in resource requirements when the availability of resources is constrained is extremely limited. It is important to consider how project duration will be affected by limited resources due to the fact that many projects must be completed with a limited set of resources. Resource constraints can be a key factor determining the project duration, and the duration can vary greatly with different levels of resource availability.

Consideration of resource allocation has emerged as an important aspect of project scheduling. Many researchers have worked on methods to minimize project duration given a limited set of resources. Ozdamar and Ulusoy [1995] extensively surveyed the approaches used to solve the resource constrained project. The goal of this paper is to demonstrate the application of a simple algorithm which can be easily applied to various kinds of resource-constrained, randomized activity duration project scheduling problems, and that will perform better in most cases than existing heuristics. The application of an optimization method called Simulated Annealing (SA) is developed in this paper. SA has been applied to many difficult optimization problems, such as the traveling salesman problem, computer (VLSI) design and graph partitioning problem, and good solutions have been obtained.

## **Methodology**

As stated earlier, many heuristic rules have been developed to solve resource constrained project scheduling problems such as the one described in Figure 1 and Table 1. The heuristics basically assign priorities to each of the activities based upon some rule. Then, for a given day  $k$  (or time unit  $k$ ) the activity with the highest priority is scheduled first if its

precedence requirements have been met and the available resources are sufficient for day  $k$ . When there are not enough free resources available for day  $k$ , day  $k+1$  is considered. Typically, ties between activities which have the same priority are broken arbitrarily. Also, most algorithms require that once an activity is started it will not be preempted.

In the SA approach that will be discussed later, two heuristics were used to provide initial feasible solutions. SA is then used to improve upon the initial solutions provided. The Composite Allocation Factor (CAF) method, proposed by Badiru [1991] and the Minimum Slack First (MINSLK) method, used by Morse, McIntosh and Whitehouse [1996] were chosen to provide initial feasible solutions.

**Application of Simulated Annealing to the Deterministic Problem.** SA is used to search for the best schedule, or the schedule that minimizes the total time required to complete a project given precedence requirements and resource constraints. The set of all feasible schedules which satisfy the precedence relationships and resource constraints is a finite set of unknown size and is denoted as  $S$ . A function  $f$ , a real valued function, is defined as the project duration on the schedules of set  $S$ . The goal is to find a feasible schedule from set  $S$  which minimizes or nearly minimizes  $f$  over set  $S$ .

A pseudo-code for the SA algorithm based upon the implementation found in Eglese [1990] is stated as follows (new terms are defined following the code):

- Step 1.** Select an initial feasible schedule, denoted as  $S_i$ .
- Step 2.** Set temperature change counter  $t = 0$ .
- Step 3.** Select an initial temperature  $T(0) > 0$ .
- Step 4.** Set repetition counter  $n = 0$ .
- Step 5.** Generate a feasible schedule  $S_j$  from the neighborhood of schedule  $S_i$ .
- Step 6.** Calculate  $\delta = f(S_j) - f(S_i)$ .
- Step 7.** If  $\delta < 0$  then  $S_i$  is set equal to  $S_j$ ;  
           else if  $\text{random}(0, 1) < \exp(-\delta / T(t))$  then  $S_i$  is set equal to  $S_j$ ;  
           otherwise,  $S_i$  remains unchanged.
- Step 8.**  $n = n + 1$ .
- Step 9.** If  $n < N(t)$  then go to Step 5.
- Step 10.**  $t = t + 1$ .
- Step 11.**  $T(t) = \alpha T(t-1)$ .
- Step 12.** If stopping criterion is true then stop; else go to Step 4.

In application discussed in this paper, the SA technique was chosen as an improvement algorithm. Therefore, an initial feasible schedule is required before the application of SA begins. As discussed earlier, the Composite Allocation Factor rule and the Minimum Slack First rule were chosen to provide initial feasible solutions. The temperature,  $T(t)$ , is started at a relatively high value and is decreased with a proportional function,  $T(t) = \alpha T(t-1)$ , where  $\alpha$  is a constant. A certain number,  $N(t)$ , of neighborhood schedules are tried at each temperature. The neighborhood of  $S_i$  is the set of all schedules which satisfy the precedence

requirements and which differ from  $S_i$  in only two positions (i.e. the sequence of activities is different in only two positions).

**Application of Simulated Annealing to the Stochastic Problem.** Procedures described to this point have dealt with projects which have deterministic activity durations. The application of the SA algorithm will now be extended to the resource-constrained, randomized activity duration project scheduling problem. Although SA was developed for use on deterministic problems, it will be shown that it works well on this class of stochastic problems.

The random variables in this problem are the individual durations of each activity. It is common in PERT analysis to use three time estimates, an optimistic time estimate  $a$ , the most likely time estimate  $m$ , and a pessimistic time estimate  $b$ , in determining the distribution from which to sample for activity durations. The three time estimates might be estimated based on historical data, experience, customer requirements, simple forecasting or simulation. Although any distribution could have been used from which to sample for each activity, the common historical method was used in this paper. Thus, the formulas for the expected activity duration  $t_e$  and the variance of activity duration  $s^2$  are:

$$t_e = \frac{a + 4m + b}{6}$$

$$s^2 = \frac{(b - a)^2}{36}$$

The beta distribution was then chosen to model activity duration and the formulas for parameters  $\alpha$  and  $\beta$  of the beta distribution as derived by Badiru [1991] are:

$$\phi = \frac{5a - 4m - b}{a + 4m - 5b}$$

$$\alpha = \phi\beta$$

$$\beta = \frac{-(\phi^2 - 34\phi + 1)}{(\phi + 1)^3}$$

Since a project has random activity durations, the overall project duration is also a random variable. Given a particular schedule or sequence of activities  $n_1, n_2, n_3, \dots, n_{N-1}, n_N$ , the expected project duration for scheduling the activities in this order is computed as follows:

- Step 1.** Select number of repetitions  $R > 0$  (sample size).
- Step 2.** Set the repetition counter  $n = 0$ .
- Step 3.** Set summation of project duration  $SumDuration = 0$ .
- Step 4.** Generate random activity durations from the beta distribution.
- Step 5.** Compute the project duration by allocating resources according to the schedule order  $n_1, n_2, n_3, \dots, n_{N-1}, n_N$ , denoted as  $Duration$ .
- Step 6.**  $SumDuration = SumDuration + Duration$ .

**Step 7.**  $n = n + 1$ .

**Step 8.** If  $n < R$  go to **Step 4**.

**Step 9.** The expected project duration for scheduling the activities in the order  $n_1, n_2, n_3, \dots, n_{N-1}, n_N$  is  $SumDuration/R$ .

The SA algorithm used to find the solution for a deterministic activity duration schedule is modified to find the solution for randomized activity durations. Instead of using function  $f$  for deterministic activity duration, a real valued function  $g$  is introduced. The function  $g$  is defined as the expected project duration based upon a sample of size  $R$ , as defined in the above procedure. The revised SA algorithm is as follows:

**Step 1.** Select an initial feasible schedule, denoted as  $S_i$ .

**Step 2.** Set the best schedule found  $S_c = S_i$ .

**Step 3.** Set temperature change counter  $t = 0$ .

**Step 4.** Select an initial temperature  $T(0) > 0$ .

**Step 5.** Set the not found better schedule counter  $NotBetter = 0$ .

**Step 6.** Set the repetition counter  $n = 0$ .

**Step 7.** Generate a feasible schedule  $S_j$ , from the neighborhood of schedule  $S_i$ .

**Step 8.** Calculate  $\delta = g(S_j) - g(S_i)$ .

**Step 9.** If  $\delta < 0$  then  $S_i$  is set equal to  $S_j$  and  $NotBetter = 0$ ; else if  $random(0, 1) < \exp(-\delta/T(t))$  then  $S_i$  is set equal to  $S_j$  and  $NotBetter = 0$ ; otherwise,  $S_i$  remains unchanged and  $NotBetter = NotBetter + 1$ .

**Step 10.** If  $g(S_i) < g(S_c)$  then  $S_c = S_i$ .

**Step 11.**  $n = n + 1$ .

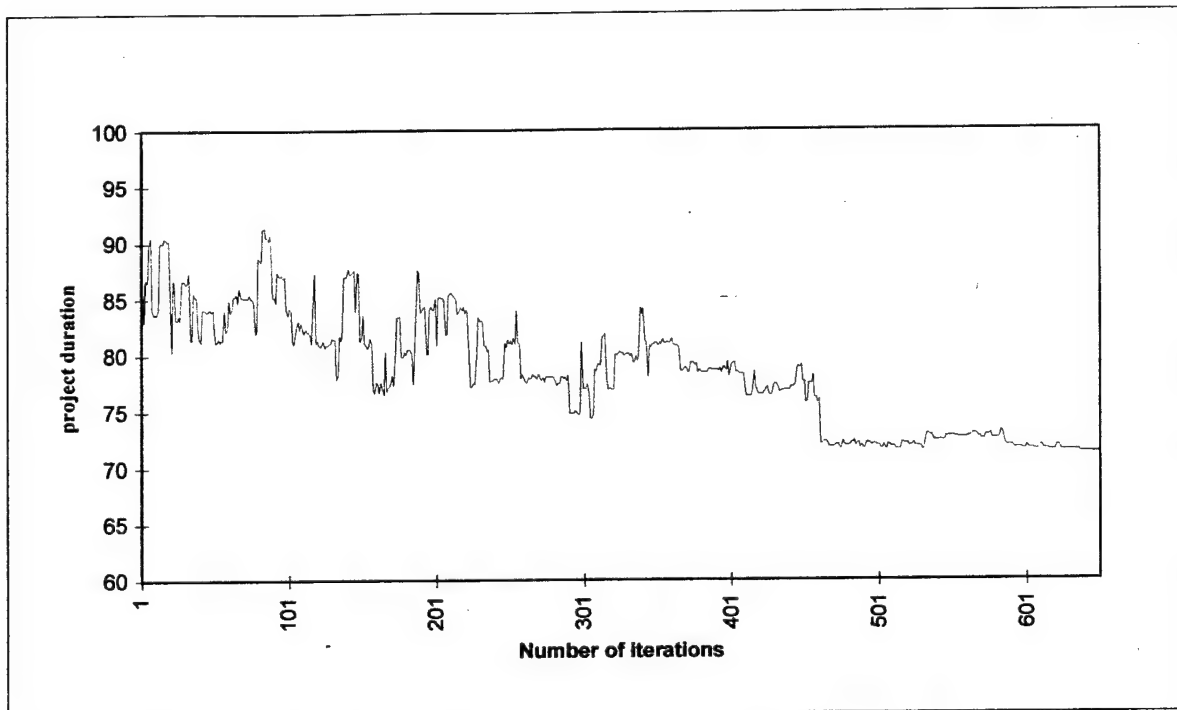
**Step 12.** If  $n < N(t)$  then go to **Step 7**.

**Step 13.**  $t = t + 1$ .

**Step 14.**  $T(t) = T(t-1)$ .

**Step 15.** If  $T(t) < MinT$  or  $NotBetter > Stopping$  then  $S_c$  is the best schedule found; else go to **Step 6**.

The results of the application of SA to a problem with 25 activities are shown in the following figure, which provides an illustration of the convergence towards a good solution.



**Duration of schedules generated during the SA process for a sample project with 25 activities.**

## **Results & Discussion**

The 110 test projects constructed by Patterson [1984] are used to evaluate the SA algorithm. Experimental results of application of the SA algorithm will now be presented in two parts: the SA algorithm for deterministic activity duration projects and the SA algorithm for randomized activity duration projects. In both cases, SA is applied to each of the 110 test projects of Patterson.

***SA algorithm for deterministic activity duration projects*** Different values of  $N(t)$  are used to evaluate the performance of the SA algorithm. Since the neighborhood schedule is generated randomly, the SA algorithm is repeated for 10 trials for each parameter setting. The results of the 10 trials with the initial feasible solution generated by the MINSLK rule are summarized in the table below. The next table shows the SA results when the initial schedule is generated using CAF.

**Experimental results of 10 trials for deterministic activity duration projects using SA algorithm. Initial schedule is generated using the MINSLK rule.**

	$N(T)=1*N^a$	$N(T)=2*N$	$N(T)=3*N$	$N(T)=5*N$	$N(T)=10*N$
Average percentage of project duration increase above optimum	0.869%	0.550%	0.428%	0.325%	0.229%
Average percentage of optimum obtained	77.00%	83.64%	86.18%	89.18%	91.91%
Average execution time in seconds	0.3173	0.6276	0.9176	1.5130	2.9731
Average standard deviation of execution time in seconds	0.3806	0.7793	1.1376	1.8552	3.6544
Number of projects in which optimum solution is found in all 10 trials	47	56	62	73	84

<sup>a</sup> N is the number of nodes or activities.

**Experimental results of 10 trials for deterministic activity duration projects using SA algorithm. Initial schedule is generated using the CAF rule.**

	$N(T)=1*N$	$N(T)=2*N$	$N(T)=3*N$	$N(T)=5*N$	$N(T)=10*N$
Average percentage of project duration increase above optimum	0.842%	0.630%	0.466%	0.353%	0.197%
Average percentage of optimum obtained	76.46%	81.64%	85.91%	88.46%	93.27%
Average execution time in seconds	0.3169	0.6108	0.9223	1.4827	2.9978
Average standard deviation of execution time in seconds	0.3737	0.7418	1.1275	1.8731	3.8133
Number of projects in which optimum solution is found in all 10 trials	47	53	64	73	80

As a comparison, the heuristics used by Morse, McIntosh and Whitehouse [1996] find only 44.44% of the optimal solutions when using a combination of five heuristics.

**SA algorithm for randomized activity duration projects.** The schedule which gives the lowest average project duration is reported as the best schedule found by the SA algorithm. Patterson's 110 test projects are again used to evaluate the performance of the SA algorithm. Instead of using the deterministic activity duration, three estimates are used in the same manner as discussed earlier. When evaluating the results of the application of SA to these randomized problems, the optimal project duration is not known. Therefore, the known deterministic optimal solution is used as an approximate lower bound from which to compare the results using SA. The results of application of SA to these random problems are shown in the following two tables.

**Experimental results of 10 trials for randomized activity duration projects using SA algorithm. Initial schedule is generated using the MINSLK rule.**

	$N(T)=1*N$	$N(T)=2*N$
Average percentage of project duration increase above optimum*	3.40%	2.27%
Average percentage of project duration decrease from the initial schedule duration	8.64%	9.62%
Number of projects in which project duration decreases from the initial schedule duration over 10%	45	48
Number of projects in which project duration decreases from the initial schedule duration over 15%	22	27
Number of projects in which project duration decreases from the initial schedule duration over 20%	7	11
Average execution time in seconds	10.804	21.414
Average standard deviation of execution time in seconds	9.868	19.708

\* The optimum is the optimal project duration determined with the original deterministic activity durations times 1.05.

**Experimental results of 10 trials for randomized activity duration projects using SA algorithm. Initial schedule is generated using the CAF rule.**

	$N(T)=1*N$	$N(T)=2*N$
Average percentage of project duration increase above optimum	3.36%	2.36%
Average percentage of project duration decrease from the initial schedule duration	8.55%	9.36%
Number of projects in which project duration decrease from the initial schedule duration over 10%	42	46
Number of projects in which project duration decrease from the initial schedule duration over 15%	18	20
Number of projects in which project duration decrease from the initial schedule duration over 20%	4	6
Average execution time in seconds	10.462	21.308
Average standard deviation of execution time in seconds	9.129	19.79

Almost one-half of the 110 projects have their duration decreased over 10%, and some of projects have their duration decreased by over 20% from the initial solutions obtained by utilizing only the MINSLK or CAF heuristics. On average, the overall project duration is decreased by around 9% from the initial solutions.

## **Conclusion**

In general, it has been shown that application of the SA algorithm to the resource-constrained problem provides excellent results. SA performs significantly better than application of heuristics such as MINSLK or CAF alone, or even combinations of existing heuristics. Not only does it improve on the solutions, in many instances it finds the optimal solution. And the computation time to achieve these results is relatively small. SA appears to be a powerful tool for scheduling of resource-constrained projects and could be a valuable asset to practitioners attempting to improve the average time to complete such projects.

# Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects

by Ying-Wei Tsai and Douglas D. Gemmill

Published in *European Journal of Operational Research*, **111** (1998) 129-141.

Industrial & Manufacturing Systems Engineering  
2019 Black Engineering  
Iowa State University  
Ames, Iowa 50011  
Tele: 515-294-8731  
n2ddg@iastate.edu

*The following is a condensed version of the above paper.*

## Introduction

The critical path method (CPM) and the program evaluation and review technique (PERT) were developed to solve project scheduling problems with the assumption that resource availability is unlimited. However, resource availability is limited in most situations. When the project is scheduled with a given set of resources, it is difficult to find the optimal solution. Resource constrained scheduling problems are generally NP-hard. Ozdamar and Ulusoy [1995] surveyed a wide variety of existing optimum-yielding techniques and scheduling heuristics for different types of resource constrained project scheduling problems. Because of the complexity involved in implementing the optimal techniques for large projects, the optimal techniques are not generally used in practice. Thus, many heuristics have been developed to find acceptable solutions for resource constrained scheduling problems in a reasonable amount of computation time. However, the solution quality obtained using heuristic algorithms can vary from optimal to poor. In this paper a heuristic procedure, tabu search, is applied to the resource constrained scheduling problem in an attempt to improve the solution quality over existing heuristic algorithms. Tabu search is applied to the resource constrained scheduling problems for both deterministic and stochastic activity duration. The procedures are described in the following sections.

## Methodology

Before the scheduling process is started, the relative priority ranking of all activities is done. Usually the priority ranking is calculated with some heuristic rules or formulae. Two heuristics rules are employed in this paper. The minimum slack (MINSLK) rule and the composite allocation factor (CAF) rule. Resources are allocated based on the priority ranking of activities. A feasible activity is an activity for which the required preceding activities have been completed. At each scheduling instant, only the feasible activities are considered for resource allocation. At time  $k$ , the feasible activity with the highest priority is scheduled if the

available resources are sufficient. When the number of resources available is inadequate for the next feasible activity at time  $k$ , then time  $k + 1$  is considered. Note that activity preemption and partial resource assignments are not allowed.

**Notation, Variables and Operations for Tabu Search.** Let  $N$  be the number of activities in a project,  $I = \{1, 2, \dots, N\}$  be the set of all activities, and  $\Pi$  be the set of all permutations defined on  $I$ . Any permutation  $\pi \in \Pi$  is defined as an  $N$  vector  $(\pi(1), \pi(2), \dots, \pi(N))$ . When the order of activities in a permutation  $\pi$  is consistent with the precedence relationships in the project, the permutation  $\pi$  is called a feasible sequence. That means activities can be completed in the same order as in the sequence. A permutation is not necessarily a feasible schedule. Let  $F$  be the set of all feasible sequences, then  $F$  is a subset of  $\Pi$ . A project can be scheduled according to the order in a feasible sequence under the resource constraints and hence the project duration can be determined. The goal of tabu search is to find a feasible sequence in  $F$  which gives the optimal or near-optimal schedule.

A *starting solution* is a feasible sequence obtained from a heuristic approach and the starting solution will be improved by the tabu search algorithm. Since quality of a solution obtained by tabu search is considerably affected by the starting solution, we use the MINSLK rule and the CAF rule to determine different starting solutions. A *move* is a transition from one feasible sequence to another by interchanging the positions of two activities  $i$  and  $j$ . An *objective function*  $f$  is defined as the project duration of a feasible sequence. The *value of a move* is the difference between the *objective function* value of the feasible sequence yielded by making the interchange and the *objective function* value of the current feasible sequence. If the *value of a move* is negative, then the *move* is called an *improvement move*.

The tabu list is used to prevent a solution from being revisited for a certain number of iterations. In this study, two tabu lists are implemented due to the different characteristics of an activity. The activities of a project are divided into two categories: *critical activity* and *non-critical activity*. An activity is classified as a *critical activity* if it is on the critical path of the project scheduled using CPM/PERT. Otherwise it is a *non-critical activity*. A list named *TabuListC* consists of *critical activities* and another list named *TabuListNC* consists of *non-critical activities*.

When resource constraints are included, in general when a *critical activity* is delayed, the successors of the delayed activity will also be delayed. Consequently, the completion time of the resource constrained project will be delayed with a high probability. For the *critical activity*, once it is moved forward due to an interchange it can be completed earlier. It is then recorded in *TabuListC* and will remain there for *TabuTenureC* iterations. While a *critical activity* is in *TabuListC*, it is in *tabu status* and *TabuTenureC* is called the *tabu tenure* of a *critical activity* which is the duration the *critical activity* remains in *tabu status*. *TabuListNC* and *TabuTenureNC* are implemented in the same manner except that the *non-critical activity* is not moved forward in *TabuTenureNC* iterations.

An integer *NumOfMove* is the number of *candidate moves* which are generated and evaluated in each iteration. These *candidate moves* are recorded in a *candidate list*. Each *candidate move* in the candidate list is evaluated by an *evaluation function*. The *evaluation function* is a combination of the *value of a move* and a *penalty function*. A *penalty function* is

applied when some predefined constraints are violated. Since only feasible sequences are considered in our scheduling application, no *penalty function* is needed.

When a feasible sequence yielded by performing a *move* contains a tabu activity, the *move* is called a *tabu restricted move* and an *aspiration test* on this *move* must be made. The *aspiration test* is an important element of flexibility in tabu search. The tabu status can be overruled if the *aspiration test* is satisfied, otherwise the *move* is not permitted. The *aspiration test* used in this paper is as follows: if the project duration of the new sequence is shorter than the best project duration found thus far, then a tabu status can be overruled. An *admissible move* is a *move* which is not a *tabu restricted move* or a *tabu restricted move* which satisfies the *aspiration test*. A *best move* is the *admissible move* that has the best value when evaluated by the *evaluation function*. The sequence yielded by making the *best move* serves as the starting solution of the next iteration. The best sequence is not necessarily an improvement over the present solution. This is the technique that tabu search employs to escape from a local minimum.

The performance of tabu search depends on several parameters. One of the most important parameters is the *tabu tenure*. The preferred choice for the value of tabu tenure is customarily based on an empirical test. A small tabu tenure will very likely introduce cycling into the search, and if the tabu tenure is too large the tabu search will be restricted to a small domain. Tabu tenure values for a scheduling problem typically can be expressed as a simple function of the number of activities, such as the square root of the number of activities. Another important parameter is the number of *candidate moves*, *NumOfMove*, considered in each iteration. Both solution speed and quality can be significantly influenced by the use of appropriate candidate list strategies.

***Application of Tabu Search to the Deterministic Problem.*** Tabu search for a resource constrained, deterministic activity duration project scheduling problem is implemented as follows:

**Step 1.** Select a starting feasible sequence, denoted as  $S_i$ .

- The starting solution can be determined using a heuristic rule, such as the MINSLK rule.
- Let the best sequence found so far, denoted as  $S_b$ , be the feasible sequence  $S_i$ .

**Step 2.** Divide activities into two parts: *critical activity* and *non-critical activity*.

- Schedule the project, without considering the resource requirements, using CPM.
- Identify the *critical activities* and the *non-critical activities*.

**Step 3.** Initialize variables used in tabu search.

- Construct the *precedence matrix* for the resource constrained project.
- Let the lists *TabuListC* and *TabuListNC* be empty lists.
- Select values for *TabuTenureC* and *TabuTenureNC*.
- Let the elements of the vectors *TabuStatusC* and *TabuStatusNC* be zeros.
- Select value for *NumOfMove*, the number of *moves* in the *candidate list*.
- Select values for stopping criteria *MaxTryOnAdmissible* and *MaxTryOnBetter*.

- Reset value of *NotFindAdmissible*, the number of iterations in which no *admissible move* is found, and value of *NotFindBetter*, the number of iterations in which no feasible sequence better than  $S_b$  is found, to zero.

**Step 4.** Create a *candidate list* which consists of *NumOfMove* moves.

- Two activities are randomly selected and the positions of these two activities are interchanged.
- If the generated sequence is not feasible, select two other activities.
- Repeat the procedure until *NumOfMove* moves are found.

**Step 5.** Choose the best admissible candidate move.

- Let *FindAdmissible* and *FindBetter* be false.
- For each *candidate move* in the candidate list do
  - Let  $S_j$  be the feasible sequence after making the *move* on  $S_i$ .
  - Evaluate the *value of a move*,  $f(S_j) - f(S_i)$ .
  - If the *move* yields a better value than all other *moves* found admissible so far in the *candidate list* then
    - Check tabu status of the *move*. If the activity moved back is a *critical activity* and is in *TabuListC* or the activity moved forward is a *non-critical activity* and is in *TabuListNC* then the *move* is *tabu restricted*.
    - If this *move* is not *tabu restricted* then
      - Accept the *move* as best admissible candidate move, denoted as  $M_b$ .
      - Change the value of *FindAdmissible* to true.
      - Reset *NotFindAdmissible* to zero.
    - Else
      - The *move* passes the *aspiration test* if  $f(S_j) < f(S_b)$ .
      - If this *move* passes the *aspiration test* then
        - Accept the *move* as best admissible candidate move, denoted as  $M_b$ .
        - Change the value of *FindAdmissible* to true.
        - Reset *NotFindAdmissible* to zero.
    - Endif
  - Endif
  - Enddo

**Step 6.** Make the best admissible candidate move.

- Let  $S_j$  be the feasible sequence after making the *best move*  $M_b$  on  $S_i$ .
- If  $f(S_j) < f(S_b)$  then  $S_b$  is replaced by  $S_j$  and change the value of *FindBetter* to true.
- If *FindAdmissible* is not true, then  $\text{NotFindAdmissible} := \text{NotFindAdmissible} + 1$ .
- If *FindBetter* is not true, then  $\text{NotFindBetter} := \text{NotFindBetter} + 1$ .

**Step 7.** Check stopping criteria.

- If  $NotFindAdmissible \geq MaxTryOnAdmissible$  or  $NotFindBetter \geq MaxTryOnBetter$  then  $S_b$  is reported as the solution, else go to Step 8.

**Step 8.** Update tabu restrictions and aspiration criteria and then go to Step 4.

- Update the lists *TabuListC* and *TabuListNC*.
- Update the vectors *TabuStatusC* and *TabuStatusNC*.

The path that tabu search follows depends significantly on the starting solution. Therefore, the solution determined using tabu search also depends on the starting solution. In order to explore diversified paths so that a better solution may be found, one can repeat the procedure described above several times with different starting solutions.

**Application of Tabu Search to the Stochastic Problem.** The existing optimal-yielding techniques, such as integer programming, are applicable only to deterministic scheduling problems. However, in most situations the exact activity duration it is not known before the activity is actually accomplished. Using the expected activity duration only and applying techniques for deterministic problems to schedule the project ignores possible effects due to the randomness of the project. With a modification on the *evaluation function*, the tabu search algorithm for a deterministic problem can easily be extended to the stochastic problem. Instead of using the expected activity durations for determining overall project length, one can draw the activity durations from the historical data or a probability distribution. In this paper, we chose a beta distribution to model activity durations, although the procedure could be used for any distribution.

The three time estimates, an optimistic time estimate  $a$ , the most likely time estimate  $m$ , and a pessimistic time estimate  $b$ , used in PERT are used to calculate the parameters for the beta distribution. The formulae for parameters  $\alpha$  and  $\beta$  of the beta distribution are:

$$\begin{aligned}\phi &= \frac{5a - 4m - b}{a + 4m - 5b} \\ \alpha &= \phi\beta \\ \beta &= \frac{-(\phi^2 - 34\phi + 1)}{(\phi + 1)^3}\end{aligned}$$

Since the activity durations are randomly distributed, the overall project duration is also a random variable. Given a particular feasible sequence (generated by utilizing the expected duration of each activity and the MINSLK rule or generated by the Tabu search), the expected project duration is computed as follows:

1. One activity duration for each activity is drawn randomly from the beta distribution with the parameters calculated based on the three time estimates associated with each activity.
2. Given the feasible sequence and the randomly generated activity durations, the project duration is calculated.
3. The calculation of project duration is repeated a number of times with different sets of activity durations and then the average project duration for the particular

feasible sequence is reported as the expected project duration. For our problems we chose to use a sample size of 100.

In other words, the feasible sequence is determined using the expected durations, and then the average performance of the given sequence is determined.

## Results and Discussion

The tabu search algorithm was coded using Borland C++ Version 5.0 for a personal computer with Pentium-166 MHz CPU running under the Windows 95 operating system. All other applications are closed during tabu search program execution in order to measure the execution time of the tabu search algorithm. The 110 test projects constructed by Patterson [1984] were used to evaluate the tabu search algorithm. In addition, a comparison of tabu search is made with the results of simulated annealing [1997] applied to the same problems. Finally, three projects, MPJ1, MPJ2, and MPJ3, adopted from an actual aircraft maintenance facility were scheduled using tabu search to illustrate its ability.

The table below shows the experimental results of 10 trials on each of the 110 projects using the simulated annealing (SA) algorithm [17].  $N$  is the number of activities in a project and  $N(T)$  is the number of schedules generated at each temperature in the annealing process. "Above optimum" is the average percentage increase in project duration above the optimal project duration provided by Patterson, "optimum obtained" is the percentage of optimal solutions found, "optimum found in all trials" is the number of projects whose optimal project duration is found in all trials, and "execution time" is the average execution time for scheduling all projects in all trials. When the initial schedule is generated using the MINSLK rule, the best results occurred for  $N(T) = 10N$  where the average percentage increase in project duration above the optimum was 0.229% with an average execution time of 2.9731 seconds. Similar results were obtained when the initial solution was generated using Badiru's CAF rule.

**Experimental results of Patterson's 110 projects for deterministic problem using SA algorithm. Initial schedule is generated using the MINSLK rule.**

N(T)	N	2N	3N	5N	10N
Above optimum	0.87%	0.55%	0.43%	0.33%	0.23%
Optimum obtained	77.00%	83.64%	86.18%	89.18%	91.91%
Optimum found in all trials	47	56	62	73	84
Execution time (seconds)	0.317	0.628	0.918	1.513	2.973
Standard deviation of execution time (seconds)	0.381	0.779	1.138	1.855	3.654

Tabu search was applied to each of the 110 test projects of Patterson. Experimental results of application of the tabu search algorithm will now be presented in two parts: the tabu search algorithm for deterministic activity duration projects and the tabu search algorithm for randomized activity duration projects.

The results of the 10 trials on each of the 110 projects with the starting solution generated using the MINSLK rule are summarized in the table below. "Project duration improved" is the average percentage improvement of project duration over the initial schedule provided by MINSLK. One can easily see the experimental results are better than those of Morse *et al.* [1996]. Comparing the results given in the table above and the table below, the simulated annealing algorithm is better than tabu search when the computation time is very short, say about 0.3 seconds on average. When the average computation time is greater than 0.3 seconds, tabu search seems to have a greater ability to find the optimal project duration. Over 93% of the optimal solutions were found in these trials and the optimal solutions of 95 projects were found in all runs with an average execution time of 3.4 seconds for tabu search. This compares to over 91% of the optimal solutions found in these trials and the optimal solutions of over 84 projects found in all runs with an average execution time of 3.0 seconds for simulated annealing. Similar results were obtained when the initial solution was generated using Badiru's CAF rule.

**Experimental results of Patterson's 110 projects for deterministic problem using tabu search. Initial schedule is generated using the MINSLK rule.**

MaxTryOnAdmissible	1000	3000	6000	10000	20000
MaxTryOnBetter	100	300	600	1000	2000
Above optimum	1.40%	0.69%	0.40%	0.29%	0.19%
Project duration improved	9.29%	9.89%	10.13%	10.23%	10.32%
Optimum obtained	66.64%	81.36%	87.46%	90.64%	93.46%
Optimum found in all trials	43	62	76	87	95
Execution time (seconds)	0.283	0.687	1.173	1.853	3.396
Standard deviation of execution time (seconds)	0.353	0.813	1.303	2.074	3.694

In the application of tabu search to randomized activity duration projects, the schedule which gives the lowest average project duration is reported as the best schedule found using tabu search. When evaluating the results of the application of tabu search to these randomized problems, the optimal project duration is not known. Therefore, the known deterministic optimal solution times 1.05 is used as an approximate lower bound from which

to compare the results using tabu search. The following table shows the results using the simulated annealing algorithm and tabu search with different initial feasible schedules. In the table "Above approximate lower bound" is the average percentage increase in project duration above the approximate lower bound, "Project duration improved" is defined as before, "Improve over 10%" is the number of projects in which the value of "project duration improved" is over 10%, "Adm" is the value of MaxTryOnAdmissible, and "Better" is the value of MaxTryOnBetter.

**Scheduling results of Patterson's 110 projects for randomized problem using SA and tabu search. Initial schedule is generated using the MINSLK rule.**

	Simulated Annealing		Tabu Search	
	N(T)=N	N(T)=2N	Adm=25 Better=250	Adm=50 Better=500
Above approximate lower bound	3.40%	2.27%	3.71%	2.54%
Project duration improved	8.64%	9.62%	8.52%	9.53%
Improved over 10%	45	48	43	47
Improved over 15%	22	27	21	27
Improved over 20%	7	11	6	8
Execution time (second)	10.804	21.414	5.834	11.290
Standard deviation of execution time (second)	9.868	19.708	4.453	8.823

A portion of the experimental results of the three aircraft maintenance projects are given in the following tables. Project MPJ1 has 107 activities, project MPJ2 has 162 activities, and MPJ3 has 168 activities. MINSLK duration is the project duration determined using the MINSLK rule. CAF duration is the project duration determined using the CAF rule. Improved duration is the project duration determined using the SA algorithm or tabu search.

**Scheduling results of three projects for deterministic problem using tabu search. Initial schedule is generated using the CAF rule.**

MaxTryOnAdmissible MaxTryOnBetter	100 1000			200 2000		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	251	234	478	251	234	478
Improved duration	198.4	183.1	373.6	194.6	180.6	360.9
Project duration improved	20.96%	21.75%	21.84%	22.47%	22.82%	24.50%
Execution time (second)	3.86	22.69	10.91	7.14	22.51	18.95
Std dev of exec time	1.29	13.89	2.24	3.04	8.29	6.70

**Scheduling results of three projects for randomized problem using SA algorithm. Initial schedule is generated using the MINSLK and CAF rule.**

N(T) = N	MINSLK			CAF		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	246.45	255.01	455.43	268.50	248.71	506.60
Improved duration	215.90	201.86	403.60	216.27	204.93	405.03
Project duration improved	12.40%	20.84%	11.38%	19.45%	17.60%	20.05%
Execution time (second)	395.62	740.69	989.53	390.45	706.21	1051.0
Std dev of exec time	30.99	59.62	81.32	36.56	49.06	72.35

**Scheduling results of three projects for randomized problem using tabu search. Initial schedule is generated using the MINSLK rule.**

MaxTryOnAdmissible MaxTryOnBetter	25 250			100 1000		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	246.59	253.85	456.28	246.76	254.54	456.51
Improved duration	214.25	203.03	392.83	209.33	194.51	377.31
Project duration improved	13.12%	20.02%	13.91%	15.17%	23.58%	17.35%
Execution time (second)	64.14	134.60	224.03	235.50	406.57	640.99
Std dev of exec time	28.41	40.86	71.23	62.71	124.73	195.54

**Scheduling results of three projects for randomized problem using tabu search. Initial schedule is generated using the CAF rule.**

MaxTryOnAdmissible MaxTryOnBetter	25 250			100 1000		
Project	MPJ1	MPJ2	MPJ3	MPJ1	MPJ2	MPJ3
MINSLK duration	267.72	249.58	506.61	268.97	249.92	507.74
Improved duration	224.05	223.73	424.84	216.70	205.70	404.46
Project duration improved	16.31%	10.36%	16.14%	19.43%	17.69%	20.34%
Execution time (second)	108.73	117.41	220.46	319.96	556.82	768.31
Std dev of exec time	36.53	33.94	79.06	94.59	208.61	278.25

## **Improving Resource-Constrained Project Schedules with Look Ahead Techniques**

by Douglas D. Gemmill and Michelle L. Edwards

Published in *Project Management Journal*, September 1999, Vol. 30, No. 3, 44-55.

Industrial & Manufacturing Systems Engineering  
2019 Black Engineering  
Iowa State University  
Ames, Iowa 50011  
Tele: 515-294-8731  
n2ddg@iastate.edu

*The following is a condensed version of the above paper.*

### **Introduction**

Methods to improve the present scheduling practices were originally reported in Tsai, [1996]. As a result of further study of the depot maintenance problem, an additional method, which we refer to as "look ahead," has been developed which can provide an improved sequence in which to perform the activities of a resource-constrained project schedule.

The majority of the methods in the literature for resource-constrained problems assign resources based upon the priority values of the activities that can be scheduled. Various heuristics are used to determine the priority values. Only the activity with the highest priority is considered for resource allocation. If the resource demand of the highest priority activity exceeds the quantity of available resources, that activity, as well as all of those with a lower priority, must wait to be scheduled. The method we propose applies a "look ahead" technique that considers scheduling lower priority activities when there are insufficient resources available to begin the top priority activity. This method will schedule activities according to a heuristic rule until it is detected that a sufficient amount of resources is not available to start the activity of highest priority. In previously employed techniques, the clock is advanced to the earliest time at which this activity can begin; therefore, delaying the start of all activities with a lower priority value. The method we propose can decrease the project makespan by considering scheduling activities of lower priority.

### **Look Ahead Technique**

Optimal solutions of resource-constrained problems can be difficult to find. This is especially true of large project networks. Consequently, researchers began to consider the use of heuristic methods to solve the resource-constrained problem, as opposed to exact methods. These heuristic procedures may not find the optimal sequence of activities in every case, but they do find a feasible, "relatively good," sequence. The motivation behind the development of the techniques proposed in this paper is the improvement of the performance of existing heuristics. The objective throughout will be to minimize the project duration (makespan).

Traditional scheduling techniques such as CPM/PERT are very efficient at determining critical activities when resources are assumed to be unlimited. In contrast, analyzing a project under this assumption leads to invalid results when, in reality, the supply of resources is finite. Until recently, the consequences of resource relationships among activities have been somewhat overlooked. As discussed in our previous papers, Woodworth and Shanahan [1988], Bowers [1995], and Tsai [1996] introduce methods to simplify determination of the critical path when limited resources are taken into account.

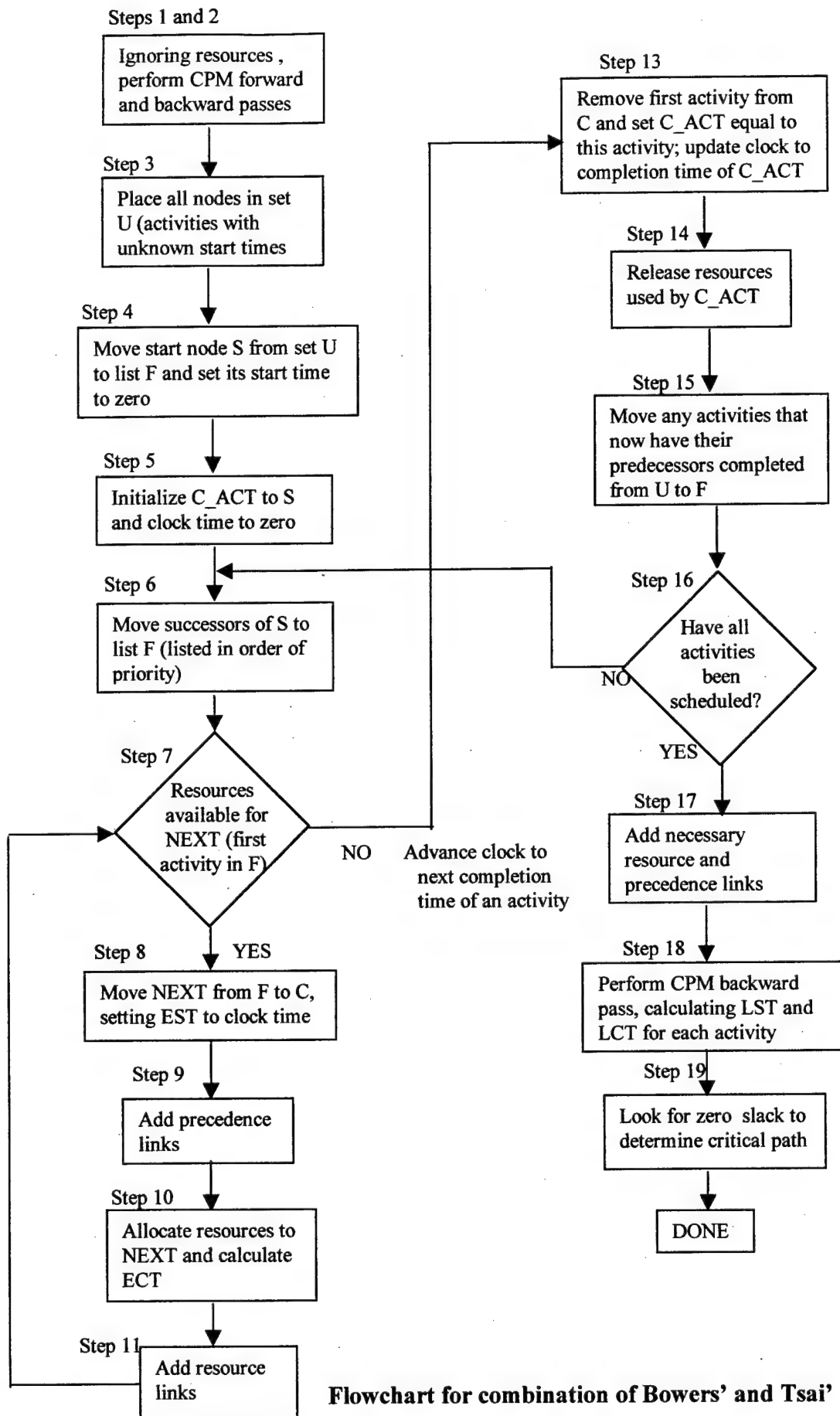
Before summarizing Bowers' and Tsai's techniques, a review of some definitions will be given. *U* is the set of all activities with unknown start times. *F* is the list of all activities, whose predecessors have all been completed, that are waiting for resources. Activities are ordered in this list by priority. *C* is the list of all activities that have been allocated resources. Here the activities are ordered by completion times. Finally, *C\_ACT* is the variable used to keep track of the most recently completed activity. Let us assume that there exists a single start node, *S*, and a single terminating node, *T*. The figure on the following page is a combination of the steps involved in Bowers' [1995] and Tsai's [1996] methods.

Once the forward and backward passes have been completed in steps one and two, one can opt to apply any of the numerous resource allocation heuristics (Ahuja *et al.*, 1993, Davis & Patterson, 1975, Morse *et al.*, 1996, Tsai, 1996) to assign activity priorities. For this demonstration, the minimum slack first (MINSLK) heuristic is employed. When assigning resources, the MINSLK heuristic gives the highest priority to the activity with the least amount of slack; ties are broken arbitrarily.

When making the forward pass through a project's CPM network, there sometimes exist enough resources to schedule a particular activity; however, since it is not the highest priority activity in list *F*, it is not considered for resource allocation at that time. This brings to mind several questions. Could one improve resource utilization by allocating free resources to lower priority activities in this case? How will the future allocation of resources be affected by scheduling a lower priority activity? Will this cause delays in critical activities? How will this influence the makespan of the entire project? The discussion of an improvement technique will begin by considering the first two of these four questions.

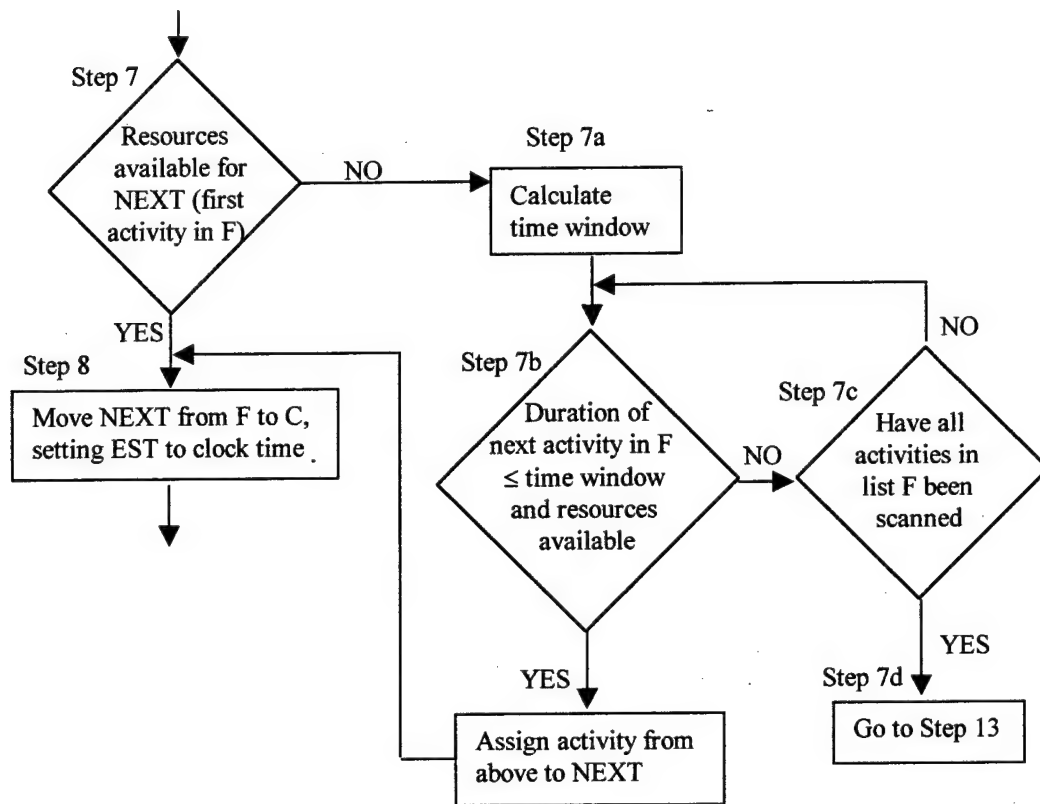
Suppose a scheduler would like to improve resource utilization by initiating as many activities at one time as is possible. Despite one's efforts to complete activities as soon as possible, however, it may be found that premature allocation of resources results in a further delay of the activity with top priority. It is feasible that the resources required for this activity will be unavailable at its previously determined EST. Attention must be directed to ensuring that these resources are available when they are needed. The remainder of this section describes a technique that looks ahead in time in an effort to determine whether or not altering the scheduling sequence will have negative effects on the resource availability and EST of the activity with highest priority.

Initially, we will only consider scheduling an activity prematurely if doing so will not cause a delay in the start of a higher priority activity. Thus, it must be able to finish before



Flowchart for combination of Bowers' and Tsai' methods.

the time at which the highest priority activity will be able to begin. An outline of this look ahead modification of Bowers' and Tsai's methods is shown in the figure below.



**Flowchart of the look-ahead process.**

The remainder of the scheduling process continues as before.

### Results obtained

Throughout this study the methods were tested using Patterson's [1984] 110 test problems. The number of activities contained in these problems ranges from five to forty-nine and the longest duration of any of these activities is nine time units. The activities of each project were prioritized using first the MINSLK heuristic and then the composite allocation factor (CAF) developed by Badiru and Pulat [1995]. In addition, the activity durations were assumed to be deterministic. The results of applying the look ahead method were compared to those obtained when strictly MINSLK or CAF was used to schedule the activities.

*Employing look ahead with MINSLK.* Testing began by applying the look ahead technique to problems in which the MINSLK heuristic was used to prioritize activities.

- Look ahead method was not used in approximately 59% (sixty-five) of the test problems.

- Look ahead was utilized once in twenty-one projects and twice in fifteen projects.
- Nine of the projects employed look ahead between three and seven times.
- In thirty of the 110 projects tested, the look ahead method caused a decrease in the makespan found by using the MINSLK heuristic alone. These improvements ranged from 2% to 13%, with an average of a 6% decrease in project duration.
- An optimal schedule was found for eighteen of the example projects.
- In seven of the 110 projects tested, the look ahead method caused an increase in the makespan. The increase in makespan ranged from 2% to 11%, with an average of 8.4 %.

*Employing look ahead with CAF.* Similar comparisons were made when the CAF method was used to prioritize activities.

- Seventy five projects did not use the look ahead method to alter the scheduling sequence.
- Twenty-six schedules were improved when the look ahead technique was applied. In this case the percentage of improvement was between 1% and 19%. The average decrease in makespan was 6%.
- Look ahead found an optimal schedule for eleven projects.
- In nine of the 110 projects tested, the look ahead method caused an increase in the makespan. The increase in makespan ranged from 2% to 22%, with an average of 6.1%.

### **Modifications of the look ahead technique**

Up to this point, the employment of the look ahead technique was allowed if and only if it did not result in a delay of the highest priority activity. Moreover, it was assumed that once an activity's priority was calculated using MINSLK or CAF, its priority value remained constant. Modifications were made to the look ahead method relative to the delay time window as well as priority values.

*Extended time windows.* Suppose that a supervisor decides that a small delay in an activity (or activities, in some cases) will not seriously affect the overall project completion. In other words, a lower priority activity could be scheduled due to look ahead, even if doing so causes the highest priority activity to be delayed by a small amount of time. Even though a critical activity may be delayed, it is possible that the overall time to complete the project will be reduced. In this situation, one could simply extend the time window by the maximum number of time units allowed.

Suppose that it is decided that permitting an activity's estimated EST to be delayed by  $x$  units of time would not have a serious effect on the entire project. Furthermore, assume that the duration of an activity is greater than the time window, but not by more than  $x$  time units. Previous conditions of the look ahead method would prevent that activity from being scheduled. However, if we define the extended time window as:

$$\text{Extended Time Window} = \text{Time Window} + x,$$

then we could allow for a delay in an activity, yet restrict the delay to no more than the amount of time permitted.

This alteration of the time window was tested in a fashion similar to that used to test look ahead previously. While an extension of the time window did not perform better than the basic look ahead method for every test problem, it proved to make significant improvements in some examples.

- The number of makespans decreased over those obtained with basic MINSLK was as high as thirty-four compared to thirty when using look ahead without extended time windows.
- Compared with basic CAF, as many as forty-seven project makespans were improved in contrast to twenty-six when using look ahead without extended time windows.
- The extended time window modification was able to reduce the project durations found by the basic look ahead technique by as much as 20%.

*Adjustment of priority values.* Consider now a possible consequence of scheduling a lower priority activity ahead of one with a higher ranking. It is possible that the successor(s) of the look ahead activity could have a higher priority value than the activities already waiting for resources. Therefore, it is possible to further delay certain activities due to altering the order in which resources are allocated. To investigate whether or not one could prevent this from happening, an adjustment was made to the priority values of certain activities in list F. Each time an activity was scheduled due to look ahead, the priority values of those activities in front of it were altered by a factor of their current priority values. Three different adjustments of the prioritization values were examined.

- Each of the values examined produced seven schedules with a worse makespan than that found by using the CAF method exclusively.
- In twenty-four problems an improvement was observed.
- Each of the values examined produced eight schedules with a worse makespan than that found by using MINSLK exclusively.
- In thirty-one problems an improvement was observed.
- The priority adjustment method was able to reduce the project durations found by the basic look ahead technique by as much as 14%.

*Combinations of the modifications.* The next logical look ahead approach to consider is a combination of the extended time windows and the adjusted priority values. One such combination, for example, would consist of extending the time window by one time unit and applying a priority adjustment factor of two. It was found that combinations of the two modifications performed better than either method did on its own. This was true regardless of whether the activities were prioritized by the MINSLK method or the CAF method. Results are summarized the results in following two tables.

### Results of applying combinations and employing MINSLK.

	# of Better Schedules <sup>1</sup>	# of Worse Schedules	Average % Improvement for Better Schedules	Range of Improvement for Better Schedules (%)	# of Optimal Schedules
Extension of 1 Time Unit					
Adjustment of 2	34	9	7	[2,25]	19
Adjustment of 3	34	9	8	[2,25]	20
Adjustment of 10	34	9	7	[2,22]	20
Changed to Neg. Value	34	9	8	[2,28]	20
Extension of 2 Time Units					
Adjustment of 2	34	9	7	[2,25]	19
Adjustment of 3	35	9	7	[2,25]	19
Adjustment of 10	35	9	7	[2,22]	19
Changed to Neg. Value	34	9	8	[2,28]	19
Extension of 3 Time Units					
Adjustment of 2	34	9	7	[2,25]	19
Adjustment of 3	35	9	7	[2,25]	19
Adjustment of 10	35	9	7	[2,22]	19
Changed to Neg. Value	34	9	8	[2,28]	19
Extension of 4 Time Units					
Adjustment of 2	35	8	7	[2,25]	19
Adjustment of 3	36	8	7	[2,25]	19
Adjustment of 10	36	8	7	[2,22]	19
Changed to Neg. Value	35	8	8	[2,28]	19
Extension of 5 Time Units					
Adjustment of 2	37	7	8	[2,25]	19
Adjustment of 3	38	7	7	[2,25]	19
Adjustment of 10	38	7	7	[2,22]	19
Changed to Neg. Value	37	7	8	[2,28]	19

<sup>1</sup> Schedules are compared to those found by employing the MINSLK method exclusively.

### Results of applying combinations and employing CAF.

	# of Better Schedules <sup>1</sup>	# of Worse Schedules	Average % Improvement for Better Schedules	Range of Improvement for Better Schedules (%)	# of Optimal Schedules
Extension of 1 Time Unit					
Adjustment of 2	36	8	6	[1,29]	13
Adjustment of 3	37	8	6	[1,29]	13
Extension of 2 Time Units					
Adjustment of 2	45	8	6	[1,29]	13
Adjustment of 3	45	8	6	[1,29]	13
Extension of 3 Time Units					
Adjustment of 2	48	9	6	[1,23]	13
Adjustment of 3	48	9	6	[1,23]	13
Extension of 4 Time Units					
Adjustment of 2	48	10	6	[1,23]	13
Adjustment of 3	48	10	6	[1,23]	13
Extension of 5 Time Units					
Adjustment of 2	49	10	6	[1,23]	13
Adjustment of 3	49	10	6	[1,23]	13
Extension of 6 Time Units					
Adjustment of 2	49	10	6	[1,23]	13
Adjustment of 3	49	10	6	[1,23]	13
Extension of 8 Time Units					
Adjustment of 2	50	10	6	[1,23]	13
Adjustment of 3	50	10	6	[1,23]	13

<sup>1</sup> Schedules are compared to those found by employing the CAF method exclusively.

It was discovered that for the problems in which MINSLK was used to prioritize activities, the combinations of extended time windows and priority adjustments found schedules with better makespans than those found with the basic look ahead method between twelve and nineteen times. In all cases, the percentage of improvement that occurred ranged from 2% to 20%, with average improvements between 6% and 8%. In the instances where the CAF method was applied, the combinations performed better than basic look ahead in nineteen to thirty-two problems. In this case, the average improvement over the basic look ahead method was between 5% and 6%. These improvements were all between 1% and 20%.

### Conclusions

The look ahead methods presented here performed quite well when tested using Patterson's (1984) 110 problems. Of the instances in which they caused a reduction in project duration, the range of the average decrease was 5% to 8% for Patterson's problems. Considering that the length of time of a typical PDM project is approximately three to six months, even a 5% decrease could mean a substantial improvement. Although the look ahead

techniques did not generate a better schedule in every case, this should not cause a problem for project schedulers. In light of the speed at which the program runs, several combinations of methods can be analyzed in a short amount of time. The software developed makes it quite simple to vary the methods used from simple MINSLK or CAF with or without look ahead to other combinations of time window extensions and priority adjustments. In fact we would recommend always beginning with simple MINSLK or CAF followed by utilization of other combinations. Overall the look ahead techniques presented show promise as an additional tool that can be used in order to reduce the total makespan of resource-constrained projects.

## **Multiple resources constrained multiple projects**

by Seong-Ryong Kim

Available as a thesis

Industrial & Manufacturing Systems Engineering

2019 Black Engineering

Major: Industrial Engineering

Iowa State University

Ames, Iowa

Tele: 515-294-8731

n2ddg@iastate.edu

*The following is a condensed version of the above thesis*

### **Introduction**

Scheduling in resource-constrained projects involves the allocation of resources to one or more activities in order to achieve a number of different objectives. The multiple resources-constrained multi-project scheduling problem can be defined as scheduling two or more projects simultaneously, such that the make span of each project is minimized. This study focuses on projects that are composed of activities which require both pooled and dedicated resources and other activities which require dedicated resources only. The objective of this study is to find a better heuristic for this class of multi-project scheduling problems.

Conflicting results in previous research simply illustrate the fact that some heuristics work more effectively in certain kinds of project scheduling problems. To cope with this feature of individual rules, we can include several priority rules in an algorithm and select the best solution among those from the rules as a final solution for the problem. Boctor [1990] suggested the use multi-heuristic procedures to get a good solution and showed a comparison of 36 small and 30 large projects. This test found that composite heuristics give better solutions than heuristics that use individual rules.

Ever since the development of critical path methods in the 1950s research has been conducted on resource-constrained multi-project scheduling problems. While there are in existence today literally hundreds of different heuristic-based scheduling rules applicable to single-projects, relatively little research has been published on rules developed specifically for the multi-project problem, and results vary widely. In this study, the *waiting count limiting method* is developed. The simulated annealing heuristic is then modified for application to this problem.

### **Methodology**

For our purposes it is assumed that all activities have a deterministic duration and

there is no preemption. That means once activities are scheduled, they cannot be interrupted by higher priority activities. Each project has both its own resources (dedicated resources) and common resources which can be shared between projects (pooled resources). The number of different resources required for each activity, and the amount of required resources are also constant.

**Proposed heuristic.** Every activity excluding the end node has some successors. If one activity has met the requirements to be scheduled, it would be desirable to be scheduled with the precedent's successors allowing some delays if that activity is a non-critical activity. Therefore, we added a value we call the "waiting count" of each activity. When a non-critical activity is delayed, even though the requirements to schedule it have been met, the activity's waiting count is incremented. When the waiting count of any activity exceeds the average number of successors for the whole project, then that activity is given the highest priority to be scheduled. Otherwise, the basic minimum slack (MINSLK) procedure is used. The proposed heuristic procedure is as follows.

- Step 1.** Calculate the earliest start time and the earliest completion time for each activity.
- Step 2.** Calculate the latest start time and the latest completion time for each activity.
- Step 3.** Compare earliest start time and latest start time, then calculate the slack of each activity.
- Step 4.** Set the current time to be 0. Initialize the resource availability.
- Step 5.** If all activities are scheduled, stop. Otherwise, place all activities whose precedence requirements have been met into "Candidate 1." If "Candidate 1" is empty, go to step 7.
- Step 5a.** Place all activities from "Candidate 1" into list "Candidate 2" whose resource requirements can be met. If "Candidate 2" is empty, go to step 7.
- Step 6.** If there are activities that have a waiting count greater than the average number of successors, go to step 6a. Otherwise go to step 6b.
- Step 6a.** Select the activity that has the greatest waiting count and assign the start time and completion time. Update the resource availability. Go back to step 5
- Step 6b.** Select an activity from "Candidate 2" with the highest priority and assign the start time and completion time. Update the resource availability. Go back to step 5.
- Step 7.** Reset the time to the earliest time at which an activity that is currently being processed will be completed.
- Step 7a.** Add 1 waiting count to the activities that are currently in the "Candidate 1." Go Back to step 5.

A sample project consisted of 30 activities (excluding the dummy start node and dummy end node) and 49 successors, so the average number of successors for this project was 1.63. Each activity can then wait for 2 counts before the activity is given the highest priority to be scheduled assuming there are sufficient resources to schedule the activity. The proposed heuristic uses MINSLK when there is a tie between the waiting count of activities, while MINSLK use the FCFS rule. In addition to the basic MINSLK heuristic

and our proposed heuristic, the simulated annealing (SA) heuristic is applied to sample projects in order to improve upon the solutions provided by the other heuristics. The MINSLK method and the proposed heuristic rule are used to provide initial feasible solutions for SA.

After an initial feasible schedule is found using the MINSLK method, a neighborhood schedule is found by interchanging two activities. The feasibility of the new schedule is checked using the precedence matrix. If infeasible, two other activities are interchanged until a feasible solution is found. The SA heuristic will accept the neighborhood schedule as a new best schedule if the makespan of neighborhood schedule is shorter than the initial feasible solution. Otherwise, SA will compute the difference between the makespan of the initial feasible solution and makespan of the neighborhood schedule, and accept with probability defined by the acceptance function  $\exp(-\delta/T(t))$  (where  $\delta = \text{makespan of the neighborhood schedule} - \text{makespan of the initial feasible solution}$ ). SA continues to evaluate new schedules in this fashion until the stopping criteria are met.

When using this method, the main makespan of some of the individual projects may be substantially increased because the acceptance of a new schedule is based only on the total makespan (makespan of all projects). Since multi-projects are composed of two or more projects and these projects are active at the same time, a different criterion is necessary to evaluate the newly generated schedules. Therefore, the SA algorithm was modified to compare the makespan of each project respectively. Focus was on each project duration, not just on the total makespan of all the projects combined. In this way each of the projects could reduce their makespan, at the same time reducing the total makespan.

The basic algorithm is almost the same except the acceptance function. After generating the neighborhood schedule, SA computes the makespan of each project. If the makespan of all individual projects is better than or equal to the makespan of the initial feasible schedule, then the newly generated schedule is accepted as a new feasible schedule. Otherwise, the improvement in the makespan of each project is computed.  $(S_{jk} - S_{ik})/S_{jk}$  (where,  $S_j = \text{makespan of newly generated schedule}$ ,  $S_i = \text{makespan of initial feasible schedule}$  and  $k = 1 \sim \text{number of project}$ ) is considered as the improvement for each project. If the sum of the improvement functions is greater than or equal to zero, then the new schedule is regarded as a new feasible schedule. Otherwise, we use the acceptance function as a criterion to accept with probability of  $\exp((\text{improvement})/T)$ .

The initial feasible schedule is stored as a best schedule. Once a newly generated schedule is accepted as a new feasible schedule, we compare that schedule to the best schedule. As suggested by Glover and Greenberg (1989), storing the best solution so far is a very efficient way to avoid the poor results. If each of the project makespans are better than or equal to the makespan of the best schedule, we accept that schedule as a new best schedule. The figure on the following page shows a flowchart of this modified SA algorithm.

Until now, the schedule that has made each project shorter than the previous one has been accepted as a best schedule. When considering a multi-project that consists of

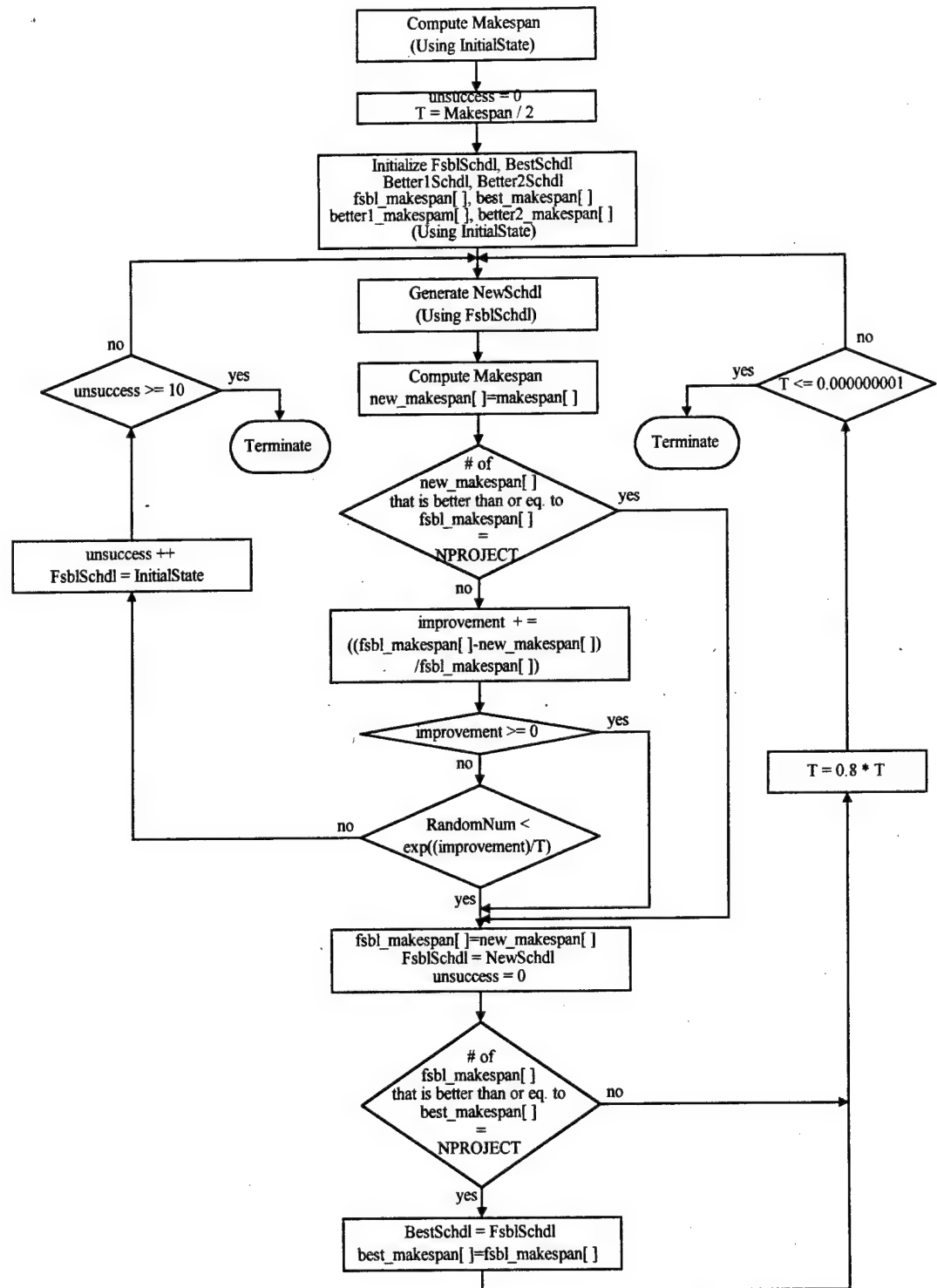
many sub-projects, there is a very low probability in accepting any schedule as the best schedule because of the low probability of reducing the makespan of each project. Therefore the evaluation function needed to be modified. The total sum of the makespans for each project could be improved even though some individual makespans may be increased. Even in the worst case, it would be possible to get an improvement in the total sum of the makespans for each project although there is only improvement in one project.

Therefore, the evaluation function is divided into three parts. The first case is to check whether the makespans of all projects have been shortened. The second case is to check whether the number of projects with improved makespans is greater than one-half of the total number of projects. The third case is when less than one-half of the project makespans have been improved.

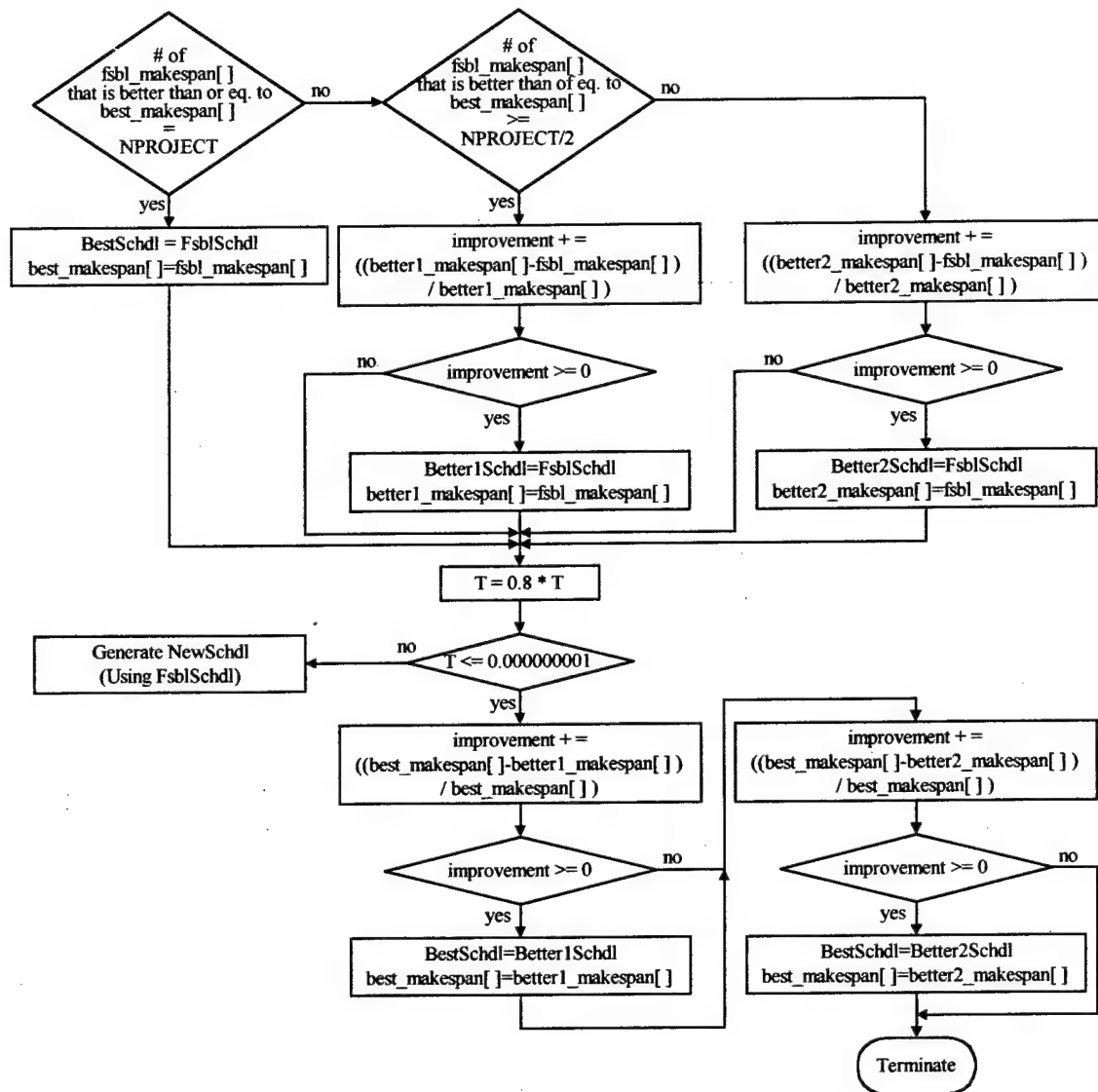
In the first case, the new schedule is considered the "Best Schedule" because all of the makespans are improved. In the second case, we perform another test to check the improvement of the sum of makespans. The difference between the "better1 makespan" of each project and the newly generated makespan of each project is divided by the "better1 makespan" of each project respectively. If that result is greater than or equal to zero, the total makespan of the whole project can be regarded as improved and can be used as a new "better1 makespan" of each project. If that result is less than zero, the generated makespan of each project is not considered as a "better1 makespan" of each project. The third case is handled in a similar fashion as the second case. When stopping criterion is met, those three schedules are compared, and the best one is selected as a new "Best Schedule". The figure below shows the flowchart of this modified evaluation function.

## **Applications and results**

The proposed and SA heuristics were applied to 90 multi-projects in which 39 projects consist of two projects, 30 projects consist of three projects and 21 projects consist of four projects. Some of the results are shown in the tables that follow. Complete results can be found in the original thesis.



Flowchart of modified SA.



**Flowchart of modified evaluation function.**

**Number of improved projects and amount of improvement when using the proposed heuristic alone.**

	2 Project	3 Project	4 Project
# of Better Schedules <sup>1</sup>	31	25	17
# of Worse Schedules	5	5	4
Average % Improvement for Better Schedules	6.6	11.1	22.5
Range of Improvement for Better Schedules (%)	0.1~34.6	0.1~27.3	1.2~43.3
# of Better Schedules above 10% improvement	7	13	11

<sup>1</sup> Schedules are compared to those found by employing the MINS�K method.

**Results of modified SA application.**

	2 Projects	3 Projects	4 Projects
# of Better Schedules <sup>1</sup>	29	9	5
# of Worse Schedules	0	0	0
Average % Improvement For Better Schedules	13.1	13.3	31.4
Range of Improvement For Better Schedules (%)	1.4~31.8	1.1~27.9	17.9~54.0
# of Better Schedules above 10% improvement	19	5	5

<sup>1</sup> Schedules are compared to those found by employing the MINS�K method.

**Comparison of MINS�K and SA with three evaluation functions.**

	2 Projects	3 Projects	4 Projects
# of Better Schedules <sup>1</sup>	33	23	13
# of Worse Schedules	0	0	0
Average % Improvement For Better Schedules	12.5	15.2	28.9
Range of Improvement For Better Schedules (%)	1.6~31.8	1.1~30.8	0.7~54.0
# of Better Schedules above 10% improvement	20	14	11

<sup>1</sup> Schedules are compared to those found by employing the MINS�K method.

Chams et al. [1987] and Johnson et al. [1987] addressed that the use of a good initial solution may improve both the quality of the solution and the total run time. Since the result of the proposed heuristic is better than the MINSLK method, we used the result of the proposed heuristic as an initial feasible solution for modified SA. As Chams et al. and Johnson et al. suggested, the result obtained from the modified SA that uses the better solution as an initial solution is better than the results obtained from the modified SA that uses the result of MINSLK method as an initial solution. The table below summarizes the comparison of the modified SA application with the use of different initial solutions.

**Comparison of modified SA heuristic using different initial solutions.**

	# of Improved Projects (%) <sup>1</sup>	
	Modified SA (MINSLK)	Modified SA (Proposed)
2 Project problems	29 (74.4)	35 (89.7)
3 Project problems	9 (30.0)	27 (90.0)
4 Project problems	5 (23.8)	18 (85.7)

<sup>1</sup> Schedules are compared to those found by employing the MINSLK method.

As with other heuristics, the proposed heuristic does not generate the best schedule in every project. However it tends to generate better schedules the majority of the time. The best results were obtained when the proposed heuristic was used to provide the initial solution for the modified SA method with three evaluation functions. Although the proposed heuristic often provided the best solution, no one method is totally dominate. Therefore, the best approach would be to use several heuristics in order to improve the chances of generating a good solution.

# Optimizing the Project Duration of Multiple Mode Resource-Constrained Projects using Simulated Annealing

by Ravikumar Rangaswamy and Douglas D. Gemmill

under review by *Project Management*

Industrial & Manufacturing Systems Engineering  
2019 Black Engineering  
Iowa State University  
Ames, Iowa 50011  
Tele: 515-294-8731  
n2ddg@iastate.edu

*The following is a condensed version of the above paper.*

## Introduction

In this research, we study the non-preemptive, resource-constrained project scheduling problem in which each activity can be executed in one of the several different execution modes. Each execution mode has a known duration and resource requirement. The objective is to determine the starting time of each activity and which execution mode should be used in order to minimize the overall project duration. The project is also described by many precedence relations and the number of resources available is limited. These constraints and resource limits should also be observed.

The acyclic activity-on-node graph is used to represent the project. Each activity has several execution modes. Each execution mode is described by a duration and the resource requirements to execute the activity in the time period. Modes having shorter duration require higher amount of resources. Two types of constraints are to be met. First, activities are subject to some precedence constraints, that is, an activity can only start after all its predecessor activities have been completed. Second, resources are available in limited quantities but renewable from period to period. It is also assumed that activities cannot be interrupted once started (no preemption is allowed). The objective is to minimize the overall project duration. To achieve this objective two decisions have to be made. First, which activity should be scheduled when and for each of those activities, which execution mode is to be used. The final schedule should contain the starting time of each activity and the execution mode to be used.

The multi-mode, resource-constrained project scheduling was first introduced by Elmaghraby [1977]. Since then the problem is studied and researched and several contributions have been published. Slowinski and Weglarz [1978] and Weglarz [1980] tried to formulate and solve the pre-emptive case where time-resource tradeoffs were described by a continuous function. Slowinski [1980, 1981] studied the case of discrete time-resource functions. Talbot [1982] formulated the not-preemptive case with discrete time-resource functions. Bector [1996] presented a heuristic procedure for the multi-mode, resource-constrained project scheduling problem in which resources are limited

but renewable from period to period. In this heuristic, some schedulable combinations of activities are enumerated and the one having the best value for an evaluating criterion is chosen. Boctor [1996] presented a general framework to solve large-scale problems. Boctor also suggested some heuristics that can be used within the suggested framework. Boctor researched the use of rules to select activities like minimum slack (MINSLK), minimum late finish time (MINLFT), minimum processing time (MINPTM), etc. Also the mode selection rules, like shortest feasible mode (SFM), least criticality ratio (LCR) and least resource proportion (LRP), were tested. It was found that the MINSLK-SFM combination produced the shortest project durations in more test cases than the other combinations. In this project, the SFM heuristic is used to select mode along with the MINSLK and CAF heuristic rule to select the activity to be scheduled next.

### **Initial solutions**

This section describes the scheduling heuristic rules used to obtain initial solutions for the multiple mode resource-constrained scheduling projects. In this research, the shortest feasible mode heuristic, proposed by Boctor [1996], is used in the project to select the mode of execution. The Composite Allocation Factor (CAF), proposed by Badiru [1991], and the minimum slack (MINSLK) heuristics, proposed by Morse, McIntosh, and Whitehouse [1996], are used to select activities to schedule.

### **General Framework**

The general framework used in this project is described as two steps:

- a. While there are still some activities to be scheduled,
  1. Move to next time period,  $t$ . The time period for the first iteration is 0.
  2. Create a list of all the activities that can be scheduled, at time period  $t$ , according to the precedence constraints and the resource requirements. If the list is empty, go to step 1.
  3. Chose an activity  $i$  from the list according to the heuristic to select activities.
  4. Chose an execution mode for the selected activity  $i$ , from all the possible execution modes of  $i$ , according to the heuristic to select the execution mode and the time period  $t$  is the earliest starting time for the activity  $i$ . The resource and precedence links are noted.
  5. Go to step 2.
- b. After all the activities are scheduled, a backward pass is run using the resource and precedence links to determine the latest start times and latest finish of each activity. Also resources are allocated during step 4 and resources that were allocated to activities that have completed are released in step 2, before the list is formed.

## **Execution Mode Selection Problem**

Boctor describes heuristics for mode selection that can be combined with heuristics to select activities to give good results for the multiple mode resource-constrained scheduling projects. Boctor describes a general framework to schedule activities in a multiple mode resource-constrained scheduling project. He also describes heuristics to select activities and modes that can be used with the suggested framework. For mode selection problem, Boctor explores three heuristics, shortest feasible mode (SFM), least criticality ratio (LCR) and least resource proportion (LRP).

The shortest feasible mode (SFM) heuristic prescribes that the shortest feasible mode is chosen. The shortest feasible mode is the mode with the shortest duration and there are resources available for its execution. Usually the shortest mode is the one which requires the maximum number of resources. Scheduling this execution mode could postpone some of the activities that could be executed in parallel to this or those that follow. This may increase the duration of the project. According to Boctor's results on 240 test problems, the SFM rules outperform significantly the LCR and the LRP rules. In our project, because of this result, we chose to implement the SFM heuristic for the mode selection problem.

## **Simulated annealing algorithms to improve the initial solutions**

The initial solutions obtained in the previous chapter are obtained by heuristics and might not be the optimal solution. In this project, simulated annealing (SA) is applied to the initial solution to improve the initial solution. Three simulated annealing algorithms are devised to improve the initial solutions. Two of these algorithms are different in the way they change the existing solution to generate new solution. The third algorithm combines the other two. Each of the algorithms is discussed in the following sections.

### ***Nested Simulated Annealing***

There are two layers of search in this implementation of the simulated annealing. The outer layer of the algorithm changes the execution mode of some randomly selected activity to some random mode of the activity in the schedule of the outer loop. This schedule is passed to the inner loop. The inner loop runs a search and returns a schedule to the outer loop. This schedule is accepted as the schedule of the outer loop if it is better than the current schedule or if the acceptance function accepts it. The best schedule of the outer loop is also updated if this schedule is better than the current best schedule. The loop continues until the stopping conditions are reached and when the search ends, the best schedule is returned as the best schedule of the project. Initially the solution provided by the heuristics is taken as the schedule and the best schedule of the outer loop.

The inner loop initializes the schedule and the best schedule of the loop with the schedule provided by the outer loop and searches its neighborhood. New neighborhood schedules are generated by selecting two activities at random and swapping their positions in the schedule of the inner loop. This schedule is evaluated and accepted as the schedule of the inner loop if it yields a better overall project duration than the current schedule or if the acceptance function accepts it. The best schedule of the inner loop is

updated if this schedule is better than the current best schedule. This search continues until the stopping conditions are reached and the best schedule is returned to the outer loop when the search stops.

The method used to determine whether or not a neighbor of  $S_{inner}$  satisfies the precedence requirements is the same as used in Tsai [1998]. Assume that the number of activities is  $N$ . An  $N$  by  $N$  precedence matrix,  $P$ , was used to verify that the schedule satisfies the precedence requirements. Each element of precedence matrix,  $P$ , is set as:

$P_{i,j} = 1$ , if node  $j$  is a successor of node  $i$ .

$P_{i,j} = 0$ , otherwise.

Suppose there is a feasible schedule sequence,

$n_1, n_2, \dots, n_{i-1}, n_i, n_{i+1}, \dots, n_{j-1}, n_j, n_{j+1}, \dots, n_N$

and two nodes,  $n_i$  and  $n_j$ ,  $i < j$ , are randomly selected and swapped. By interchanging these two activities, a new sequence,

$n_1, n_2, \dots, n_{i-1}, n_j, n_{i+1}, \dots, n_{j-1}, n_i, n_{j+1}, \dots, n_N$

is generated. To check the feasibility of the generated schedule, it is examined if

$P_{k,j} = 0$ , for  $k = i, i+1, \dots, j-2, j-1$

and

$P_{k,j} = 0$ , for  $k = i+1, i+2, \dots, j-1, j$

### ***Iterative Simulated Annealing***

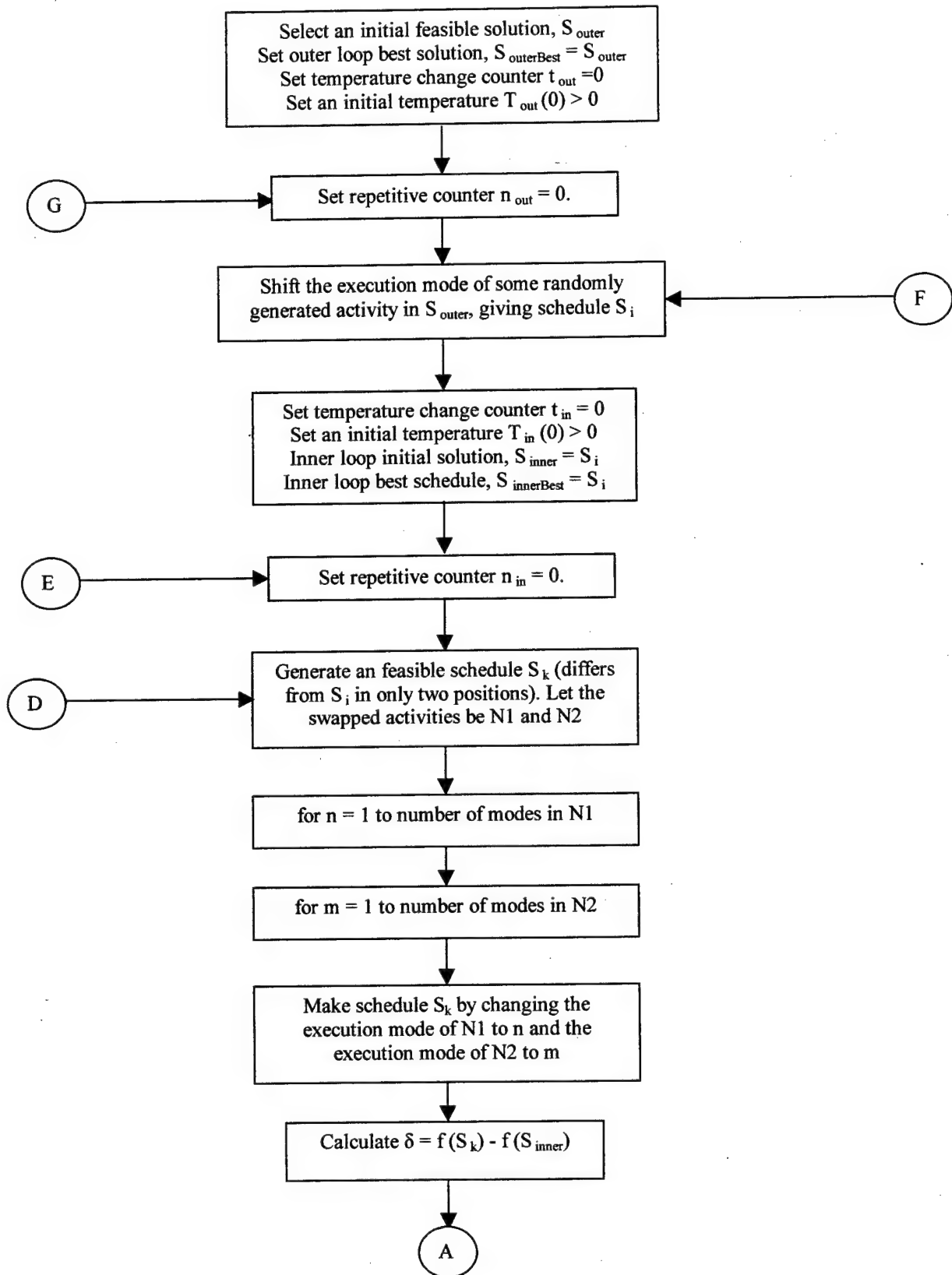
In this algorithm there is only one search loop. The feasible schedule generated by the CAF/SFM or MINSLK/SFM heuristics is taken as the initial schedule of the algorithm. This schedule is therefore taken as the current and the current best schedule of the search. New neighborhood schedules are generated by selecting two activities at random and swapping their positions in the current schedule of the algorithm. All combinations of the execution modes of the swapped activities are tried. Each of these schedules are evaluated and accepted as the current schedule of the search loop if it yields a better overall project duration than the current schedule or if the acceptance function accepts it. This schedule is also updated as the best schedule of the search if it is better than the current best schedule of the inner loop. When the search loop exits because of the stopping criteria, the best solution of the loop is returned as the best schedule of the project.

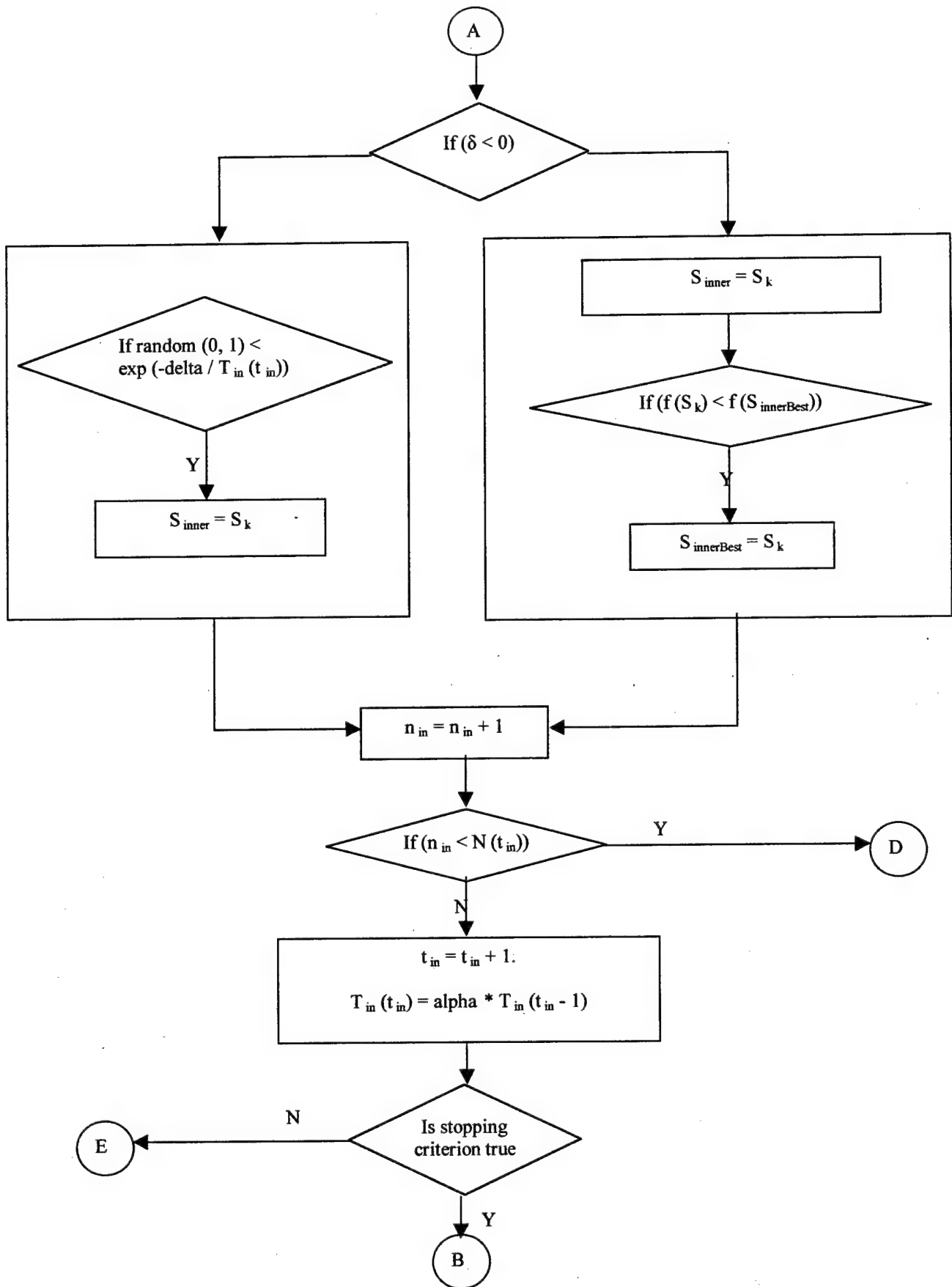
### ***Combination of Iterative and Nested Simulated Annealing***

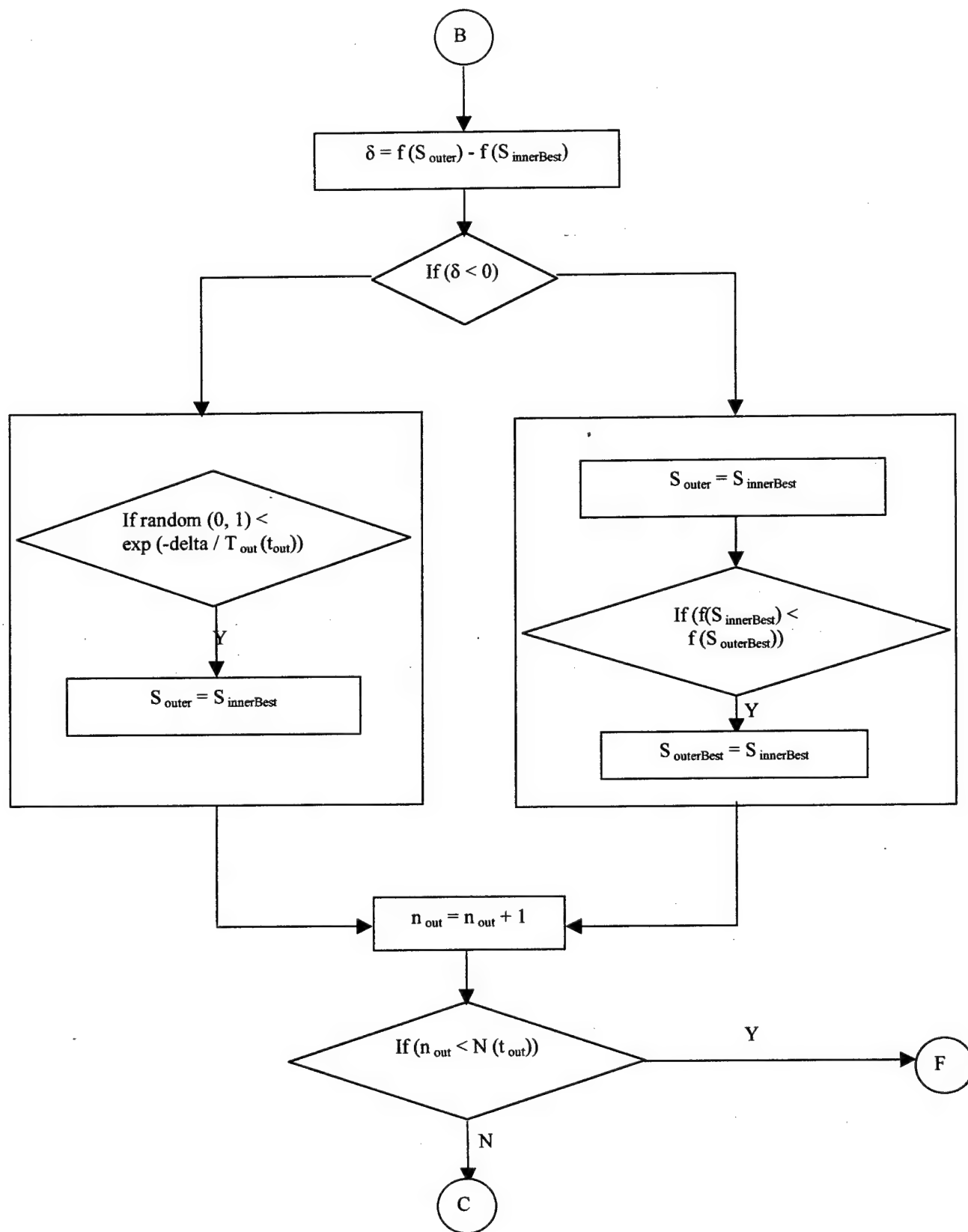
This implementation of simulated annealing is a combination of the iterative and nested simulated annealing implementations. There are two layers of search in this implementation also. As in the nested algorithm, the inner loop initializes the schedule and the best schedule of the loop with the schedule provided by the outer loop and searches its neighborhood. However, in this algorithm all combinations of the execution modes of the swapped activities are tried. Each of these schedules are evaluated and accepted as the current schedule of the inner loop if it yields a better overall project duration than the current schedule or if the acceptance function accepts it.

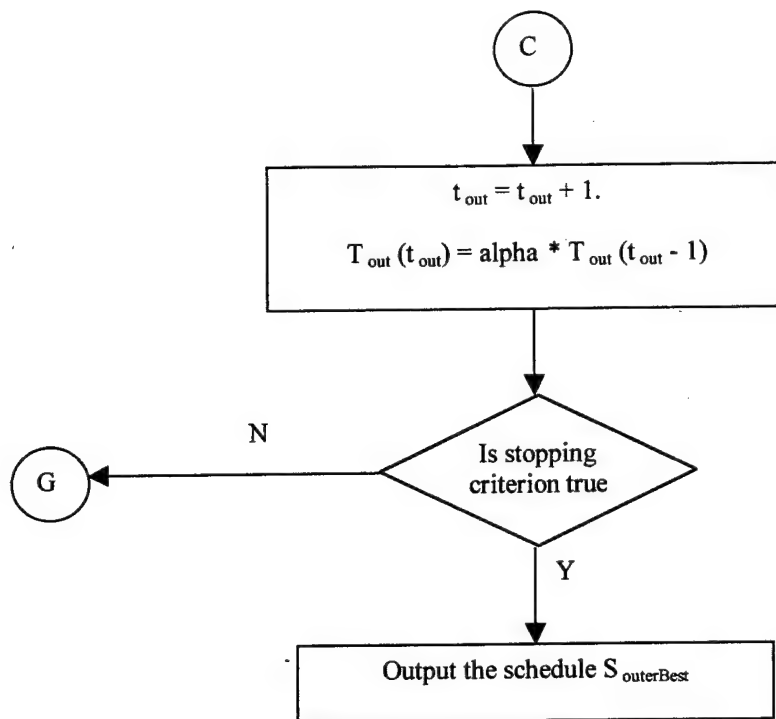
The pseudo-code for the combination algorithm is as follows:

# Combination of Nested and Iterative Simulated Annealing for Multimode Resource-Constrained Scheduling









## Results and conclusions

Test cases were generated for problems with 2, 3, 4 and 5 resource types. Twelve precedence relations were used to give a total of 48 test problems. The results are summarized in the following tables. First the MINSLK/SFM and the CAF/SFM heuristics are compared and then the three algorithms are compared.

The solutions given by MINSLK/SFM and CAF/SFM are compared in the table below. The values give the number of times best solution were given by the two heuristics at each of the 2, 3, 4 and 5 resource types and also the number of best solutions given totally. The MINSLK/SFM heuristic performed better than the CAF/SFM heuristic at each of the 2, 3, 4 and 5 resource type. Over all the test cases, MINSLK/SFM heuristic gave a best solution 75% of the time against 54% by CAF/SFM heuristic.

### Comparison of best solutions of MINSLK/SFM and CAF/SFM

Number of resource types	MINSLK/SFM		CAF/SFM	
	Number / 12	%	Number / 12	%
2	10	83.3 %	6	50.0 %
3	9	75.0 %	5	41.6 %
4	8	66.7 %	7	58.3 %
5	9	75.0 %	8	66.7 %
Total	36 / 48	75.0 %	26 / 48	54.2 %

The next table gives the comparison of the unique best solutions given by the MINSLK/SFM heuristic and the CAF/SFM heuristic. Here again the MINSLK/SFM heuristic was the better heuristic at each of the 2, 3, 4 and 5 resource types. Over all the test cases, this heuristic gave the unique best solution 46 % of the time and the CAF/SFM heuristic 25 % of the time. In the remaining 29 % of cases, both the heuristics gave the same solution.

#### Comparison of unique best solutions of MINSLK/SFM and CAF/SFM

Number of resource types	MINSLK/SFM		CAF/SFM	
	Number / 12	%	Number / 12	%
2	6	50.0 %	2	16.6 %
3	6	50.0 %	2	16.6 %
4	5	41.6 %	4	33.3 %
5	5	41.6 %	4	33.3 %
Total	22 / 48	45.8 %	12 / 48	25.0 %

The table below shows the percentage improvement by each of the simulated annealing (SA) algorithms of the solution provided by MINSLK/SFM or the CAF/SFM heuristics. The table shows the comparisons at each of the 2, 3, 4 and 5 resource types. For each of the resource types there were 12 problems tested. From the results, we could see that the iterative and the combination of iterative and nested implementations of simulated annealing were slightly better than the nested implementation.

#### Percentage improvement by the three SA algorithms

Number of resource types	Percentage Increase over the Initial Solution		
	Nested SA	Iterative SA	Combined SA
2	8.1	12.63	13.55
3	9.41	10.76	10.16
4	6.81	8.01	8.97
5	9.73	13	12.82

The following table shows the number of test problems in which each of the simulated annealing (SA) algorithms improved the initial feasible solution provided by MINSLK/SFM or the CAF/SFM heuristics. The table also shows the comparisons at each of the 2, 3, 4 and 5 resource types. For each of the resource types there were 12 problems tested. From the results, there was not a significant difference in the performance of the three implementations of the SA algorithm. Each of the algorithms improved initial solution approximately 75 % of the time.

### Number of improvements by the three SA algorithms

Number of resource types	Nested SA		Iterative SA		Combined SA	
	Number/ 12	%	Number/ 12	%	Number/ 12	%
2	8	66.7 %	10	83.3 %	10	83.3 %
3	9	75.0 %	9	75.0 %	9	75.0 %
4	9	75.0 %	9	75.0 %	9	75.0 %
5	9	75.0 %	9	75.0 %	9	75.0 %
Total	35 / 48	72.9 %	37 / 48	77.1 %	37 / 48	77.1 %

The next table shows the number of test problems in which each of the simulated annealing (SA) algorithms yielded the best schedule in terms of the overall project duration. The table also shows the comparisons at each of the 2, 3, 4 and 5 resource types. For each of the resource types there were 12 problems tested. From the results we can see that the combination of the nested and the iterative SA algorithm outperformed the nested SA and iterative SA algorithm, providing the best schedule 77 % of the time over all the test problems. The iterative SA algorithm yielded the best schedule 67 % of the time and the nested SA algorithm only provided the best schedule 50 % of the time. Except at 3 resource types, the combined SA algorithm significantly bettered the other two algorithms.

### Number of best solutions given by the three SA algorithms

Number of resource types	Nested SA		Iterative SA		Combined SA	
	Number/ 12	%	Number/ 12	%	Number/ 12	%
2	4	33.3 %	7	58.3 %	11	91.6 %
3	8	66.7 %	9	75.0 %	7	58.3 %
4	6	50.0 %	8	66.7 %	11	91.6 %
5	6	50.0 %	8	66.7 %	8	66.7 %
Total	24 / 48	50.0 %	32 / 48	66.7 %	37 / 48	77.1 %

The last table shows the number of test problems in which each of the simulated annealing (SA) algorithms yielded the unique best schedule in terms of the overall project duration. The table also shows the comparisons at each of the 2, 3, 4 and 5 resource types. For each of the resource types there were the same 12 problems tested. From the results, we can see that the combination of the nested and the iterative SA algorithm again outperformed the nested SA and iterative SA algorithm, providing the unique best schedule 18.8 % of the time over all the test problems. The iterative SA algorithm and the nested SA algorithm both provided the unique best schedule 8.3 % of the time. Except at 5 resource types, the combined SA algorithm was the best algorithm. At 5 resource types, all the three implementations of the SA algorithm gave the unique best solutions equal number of times.

### Number of unique best solutions given by the three SA algorithms

Number of resource types	Nested SA		Iterative SA		Combined SA	
	Number/ 12	%	Number/ 12	%	Number/ 12	%
2	0	0.0 %	1	08.3 %	3	25.0 %
3	1	08.3 %	1	08.3 %	2	16.6 %
4	1	08.3 %	0	0.0 %	2	16.6 %
5	2	16.6 %	2	16.6 %	2	16.6 %
Total	4 / 48	08.3 %	4 / 48	08.3 %	9 / 48	18.8 %

To summarize, the MINSLK/SFM heuristic provided the better solutions in more cases compared to the CAF/MINSLK heuristic. Of the simulated algorithm, the combination of the nested and the iterative algorithms was the best algorithm. This algorithm was also better at all levels of number of resource types and not just in the overall results.

## **Scheduling Software Developed**

Software available from Douglas D. Gemmill  
Industrial & Manufacturing Systems Engineering  
2019 Black Engineering  
Iowa State University  
Ames, Iowa 50011  
Tele: 515-294-8731  
n2ddg@iastate.edu

### **Introduction**

The objective of this research is to improve the Air Force's approach to scheduling aircraft depot maintenance. WR-ALC (Robins AFB) and a software company (Robbins Gioia) worked together to create the software that is presently utilized for scheduling, Programmed Depot Maintenance Scheduling Software (PDMSS). The PDMSS software has many strengths; however, the majority of the personnel with whom we spoke did not find the software very useful for day-to-day project scheduling. One shortcoming of PDMSS was the way in which projects were modeled. Each project was assumed to be deterministic with activity durations based on optimistic standard times rather than on the actual time it took to complete the jobs. Another weakness of PDMSS is its inability to provide an actual schedule of operations. Thus, the success of each of the maintenance projects is dependent upon the management skills of the supervisors. It was also found that the supervisors were expected to make use of PDMSS, but they were not provided with the training necessary to do so. They were required to become familiar with the system on their own. The majority found this task difficult because, in their opinion, the software was not user friendly.

The scheduling algorithms were originally coded and tested using C++ with no graphical user interface (GUI). A Windows 98 environment is being developed with the use of Borland C++ Builder to create a good GUI. To assist in project management, the windows application informs the user which activities are critical to a project's completion. More importantly, it provides a schedule of activities for the user to follow in completing the project (while attempting to minimize total makespan).

### **Current Capabilities of the Software**

The user begins a scheduling (simulation) run by specifying which files will be used by the program. Three types of files are needed: the precedence data file, the resource data file, and the output file. In the future, the user will also be able to create precedence and resource data files within an editor window if so desired. In the case where a project contains stochastic activity durations, the user must specify the number of runs to make before finding an average project duration. The default number of runs is 100. At this time, the program appends the results of each run to the output file, as specified by the user, in addition to showing the contents of this file within the main program window. The figure below shows the initial main program window, with a blank output screen.

**C-141 Project Management**

File Options Help

D:\Win Program\Ranproj1.arc  
Activity Data File

D:\Win Program\Proj1.res  
Resource Data File

D:\Win Program\Proj1.out  
Schedule Output File

**Create Schedule**

☒ **Stochastic** ☐ **Deterministic**

100  
Number of Runs

After the data files are designated, the user chooses which of the previously mentioned options will be used to schedule the project. Due to the speed of the C++ program, the user can generate and evaluate a number of schedules in a short amount of time by employing combinations of the following options:

- basic schedule determined by the MINSLK heuristic,
- basic schedule determined by the CAF heuristic,
- look ahead techniques,
- simulated annealing heuristic, and
- tabu search heuristic.

Currently, the user is able to use either MINSLK or CAF, but not both. The same is true for the tabu search and simulated annealing (SA) options. As one can see in the following figure, each of the scheduling options is included in a pull-down menu.

**C-141 Project Management**

File Options Help

☒ Minimum Slack First  
 Composite Allocation Factor   
 Tabu Search   
 Simulated Annealing   
☒ Look Ahead

D:\Win Program\Proj1.out  
Schedule Output File

☒ Stochastic 
 ☐ Deterministic

1  
Number of Runs

Create Schedule

If the user chooses an option for which one or more parameters need to be defined, a small window is activated prompting the user to supply the relative values. The user can opt to apply the default parameters if it is appropriate to do so. The window prompting the user for the parameters associated with simulated annealing is shown in the figure below.

Suppose we generated a schedule using MINSLK to prioritize the activities, applied the basic look ahead method, and used SA to search for the optimal solution. The next figure shows the current format for displaying the output. The critical column indicates an activity's criticality index, or the probability that the activity will fall on the critical path.

**Optimization Techniques** [X]

Simulated Annealing | Look Ahead

Initial Temperature: 100.0

Minimum Temperature: 0.00000000001

Number of Neighborhood Schedules Evaluated at Each Temperature: 100

Alpha: 0.8

Maximum Consecutive Iterations Without Improvement: 10

OK Cancel Help

**C-141 Project Management** [ ] [ ] [X]

File Options Help

Activity Data File: D:\Win Program\Planproj1.arc

Resource Data File: D:\Win Program\Proj1.res

Schedule Output File: D:\Win Program\Proj1.out

☒ Stochastic ☐ Deterministic

Number of Runs: 100

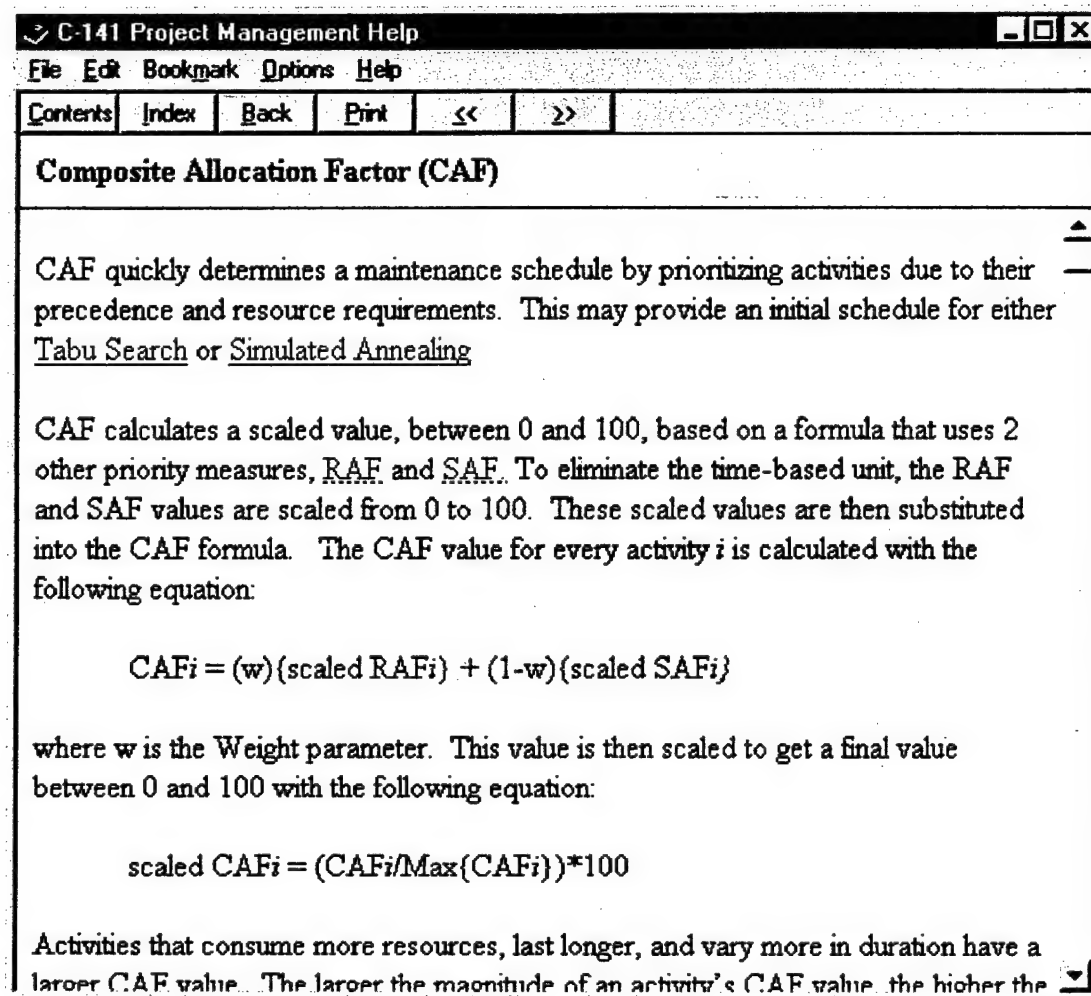
**Create Schedule**

This schedule was created with: Minimum Slack First  
Look Ahead

The makespan is 225.464

NODE	SYMBOL	DURATION	EST	ECT	LST	LCT	SLACK	CRIT
1	AA	2.126	0.000	2.126	0.000	2.126	0.000	1.00
2	AI	3.105	2.126	5.231	2.133	5.238	0.006	0.98
89	ZZ	3.193	2.126	5.320	8.612	11.806	6.486	0.00
91	PR	5.260	2.126	7.386	6.508	11.768	4.382	0.00
90	TI	31.215	2.126	33.341	8.233	39.448	6.107	0.02
3	AM	6.384	5.231	11.616	5.238	11.622	0.006	0.98
4	AN	1.048	11.616	12.664	12.843	13.890	1.227	0.00
15	BI	1.068	11.616	12.683	11.768	12.836	0.153	0.37
12	BC	2.109	11.616	13.725	11.920	14.029	0.305	0.23

We have developed some help files, and one example is given below. At this time, the capabilities of the program are still under development.



## Aggregate of References for All Papers

- Ahuja, R. K., Magnanti, T. L., & Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 993
- B. Badiru, A Simulation Approach to PERT Network Analysis, *Simulation* **57** (1991) 245-255.
- B. Badiru, P. S. Pulat, *Comprehensive Project Management*. Prentice Hall PTR, Englewood Cliffs, New Jersey, 1995.
- Bell, C. E. and Han, J., A new heuristic solution method in resource constrained project scheduling problems, *Naval research logistics*, **38** (1991) 315-331.
- Blazewicz, J., Lenstra, J. K. and Rinnoy Kim, A.H.G., Scheduling subject to resource constraints: classification and complexity, *Discrete Applied Mathematics*, **5** (1983) 11-24.
- Bock, D. B. and Patterson, J. H., A comparison of due date setting, resource assignment and job preemption heuristics for the multi-project scheduling problem, *Decision Science*, **21** (1990) 387-402.
- Boctor, F. F., Some efficient multi-heuristic procedures for resource constrained project scheduling, *European Journal of Operational Research*, **49** (1990) 3-13.
- Boctor, F. F., A New and Efficient Heuristic for Scheduling Projects with Resource Restrictions and Multiple Execution Modes, *European Journal of Operational Research*, **90** (1996) 349-361.
- Boctor, F. F., Heuristics for Scheduling Projects with Resource Restrictions and Several Execution Modes, *International Journal of Production Research*, **31** (11) (1996) 2547-2558.
- Bowers, A., Criticality in Resource Constrained Networks, *Journal of the Operational Research Society* **46** (1995) 80-91.
- Brusco, M. J. and Jacobs, L. W., A simulated annealing approach to the cyclic staff-scheduling problem, *Naval Research Logistics Quarterly*, **40** (1993) 69-84.
- Cerny, V., Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, *Journal of Optimisation Theory and Applications*, **45** (1985) 41-51.
- Chams, M., Hertz, A. and de Werra, D., Some experiments with simulated annealing for colouring graphs, *European Journal of Operational Research*, **32** (1987) 260-266.

- Connolly, D. T., An improved annealing scheme for the QAP, *European Journal of Operational Research*, **46** (1988) 93-100.
- Cooper, D. F., Heuristics for scheduling resource constrained scheduling projects: an experimental investigation, *Management Science*, **22** (1976) 1186-1194.
- Davis, E. W., Project scheduling under resource constraints: a historical review and categorization of procedures, *AIIE transactions*, **5** (1973) 297-313.
- Davis, E. W. and Heidorn, G. E., An algorithm for optimal project scheduling under multiple resource constraints, *Management Science*, **17** (1971) B803-B816.
- Davis, E. W., & Patterson, J. H., A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling, *Management Science* **21** (1975) 944-955.
- Demeulemeester, W. Herroelen, A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem, *Management Science* **38** (1992) 1803-1818.
- Demeulemeester, Minimizing Resource Availability Costs in Time-Limited Project Networks, *Management Science* **41** (1995) 1590-1598.
- B. Dreger, *Project Management - Effective Scheduling*. Van Nostrand Reinhold, New York, New York, 1992.
- Dumond, J. and Mabert, V. A., Evaluating project scheduling and due date assignment procedures: an experimental analysis, *Management Science*, **34** (1988) 101-118.
- Eglese, R. W., Simulated annealing: a tool for operational research, *European Journal of Operational Research*, **46** (1990) 271-281.
- Evans, J. R., & Minieka, E., *Optimization Algorithms for Networks and Graphs*. Marcel Dekker, Inc., New York, 1992.
- Elmaghraby, S. E., *Activity Networks: Project Planning and Control by Network Models*. Wiley, New York, 1977.
- Fendley, L. G., Toward the development of a complete multi-project scheduling system, *Journal of Industrial Engineering*, **19** (1968) 505-515.
- Gemmill, D.D. and Tsai, Y. W., Using a Simulated Annealing Algorithm to Schedule Activities of Resource-Constrained Projects, *Project Management Journal*, **28** (4) (1998) 8-20.

Glover, F. and McMillan, C., The General Employee Scheduling Problem: An Integration of Management Science and Artificial Intelligence, *Computer and Operations Research* **13** (1986) 563-593.

Glover, F. and Greenberg, H., New approaches for heuristic search: a bilateral link with artificial intelligence, *European Journal of Operational Research*, **39** (1989) 119-130.

Glover, F., Tabu Search, Part I, *ORSA Journal of Computing* **1** (1989) 190-206.

Glover, F., Tabu Search, Part II, *ORSA Journal of Computing* **2** (1990) 4-32.

Glover, F., Tabu Search: A Tutorial, *Interfaces* **20** (1990) 74-94.

Grover, L. K., A new simulated annealing algorithm for standard cell placement, *Proc. IEEE International Conference on Computer-Aided Design*, Santa Clara (1986) 378-380.

Hillier, F. S., & Lieberman, G. J. *Introduction to Operations Research*. McGraw-Hill Publishing Company, New York, 1990.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Scheveon, C., *Optimization by simulated annealing: An experimental evaluation; part 1*. AT&T Bell Laboratories, Murray Hill, NJ, 1987.

Kirkpatrick, S., Gelatt, C. and Vecchi, P., Optimization by simulated annealing, *Science*, **220** (1983) 671-679.

Koulamas, C., Antony, S.R. and Jean, R., A survey of simulated annealing applications to operations research problems, *Omega*, **22** (1994) 41-56.

Kurtulus, I. S. and Davis, E. W., Multi-project scheduling: categorization of heuristic rule performance, *Management Science*, **28**, (1982) 161-172.

Kurtulus, I. S. and Narula, S. C., Multi-project scheduling: analysis of project performance, *IIE Transactions*, **17** (1985) 58-66.

Lawrence, R., T. E. Morton, Resource-Constrained Multi-project Scheduling with Tardy Costs: Comparing Myopic, Bottleneck, and Resource Pricing Heuristics, *European Journal of Operational Research* **64** (1993) 168-187.

Li, K.Y. and Willis, R.J., An Iterative Scheduling Technique for Resource Constrained Project Scheduling, *European Journal of Operational Research* **56** (1992) 370-379.

Metropolis, N., Rosenbluth, A.N., Rosenbluth, M.N., Teller, A.H. and Teller, H., Equation of state calculation by fast computing machines, *Journal of Chemical Physics*, **21**(6) (1953) 1087-1092.

- Morse, C., J. O. McIntosh, G. E. Whitehouse, Using Combinations of Heuristics to Schedule Activities of Constrained Multiple Resource Projects, *Project Management Journal* March (1996) 34-40.
- Osman, I. H. and Potts, C. N., Simulated annealing for permutation flow-shop scheduling, *Omega*, **17** (1989) 551-557.
- Ozdamar, G. Ulusoy, A Survey on the Resource-Constrained Project Scheduling Problem, *IIE Transactions*, **27** (1995) 574-586.
- Patterson, J. H., Alternative method of project scheduling with limited resources, *Naval Research Logistics Quarterly*, **20**(4) (1973) 767-784.
- Patterson, J. H. and Huber, W. D., A horizon-varying, zero-one approach to project scheduling, *Management Science*, **20** (1974) 990-998.
- Patterson, J. H. and Roth, G. W., Scheduling project under multiple resource constraints: a 0-1 programming approach, *AIIE Transactions*, **8** (1976) 449-455.
- Patterson, A Comparison of Exact Procedures for Solving the Multiple Constrained Resource Project Scheduling Problem, *Management Science* **30** (1984) 854-867.
- Patterson, J. H., Slowinski, R., Talbot, F. B. and Weglarz, J., An algorithm for a general class of precedence and resource-constrained scheduling problems, in *Advances in Project Scheduling*, Slowinski, R. & Weglarz, J. (eds). Elsevier, Amsterdam (1989) 3-28.
- Pinedo, M. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, Englewood Cliffs, NJ, 1995.
- Potts, C. N. and Van Wassenhove, L. N., Single machine tardiness sequencing heuristics, *IIE Transactions*, **23** (1991) 346-354.
- Pritsker, A. A. B., Watters, L. J. and Wolfe, P., M. Multi-project scheduling with limited resources: a zero-one programming approach, *Management Science*, **16** (1969) 93-108.
- Punnen, A.P. and Aneja, Y.P., Categorized Assignment Scheduling: A Tabu Search Approach, *Journal of the Operational Research Society* **44** (1993) 673-679.
- Punnen, A.P., and Aneja, Y.P., A Tabu Search Algorithm for the resource Constrained Assignment Problem, *Journal of the Operational Research Society* **46** (1995) 214-220.
- Skorin-Kapov, F., Tabu Search Applied to Quadratic Assignment Problem, *ORSA Journal of Computing* **2** (1990) 33-45.

Slowinski, R. and J. Weglarz, Solving the General Project Scheduling Problem with Multiple Constrained Resources by Mathematical Programming. *Proceedings 8<sup>th</sup> IFIP conference on Optimization Techniques*. Lecture Notes in Control and Information Sciences. 7 (1978) 278-289.

Slowinski, R., Two Approaches to Problems of Resource Allocation among Project Activities: a Comparative Study, *Journal of Operational Research Society*, 31 (8) (1980) 711-723.

Slowinski, R., Multipleobjective Network Scheduling with Efficient Use of Renewable and Non-renewable Resources, *European Journal of Operational Research*, 7 (3) (1981) 265-273.

Stinson, J. P., Davis, E. W. and Khumawala, B. W., Multiple resource-constrained scheduling using branch and bound, *AIIE Transactions*, 10 (1978) 252-259.

Talbot, F. B., Resource constrained project scheduling with time-resource tradeoffs: the nonpreemptive case, *Management Science*, 28 (1982) 1197-1210.

Talbot, F. B. and Patterson, J. H., An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems, *Management Science*, 24 (1978) 1163-1174.

Thesen, A., Heuristic scheduling of activities under resource and precedence restrictions, *Management Science*, 23 (1976) 412-422.

Tsai, Y., Resource constrained project scheduling using simulated annealing and tabu search. M.S. Thesis, Iowa State University, 1996.

Tsai, Y. W. and D. D. Gemmill, Identifying the Critical Path in Resource-Constrained Projects, Working paper # 96 - 131, Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, Iowa (1996).

Y. W. Tsai and D. D. Gemmill, Using Tabu Search to Schedule Activities of Stochastic Resource-Constrained Projects, *European Journal of Operation Research*, 111 (1) (1998) 131-141.

Tsubakitani, S. and Deckro, R. F., A heuristic for multi-project scheduling with limited resources in the housing industry, *European Journal of Operational Research*, 49 (1990) 80-91.

Van Laarhoven, P.J.M., Aarts, E.H.L. and Lenstra, J.K., Job shop scheduling by simulated annealing, *Operations Research*, 40 (1992) 113-125.

Watkins, Kevin, *Discrete Event Simulation in C*, London, England: McGraw-Hill, 1993.

Weglarz, J., Control in Resource Allocation Systems. *Foundation of Control Engineering*. 5 (3) (1980) 159-180.

Wiest, J. D., A heuristic model for scheduling large projects with limited resources, *Management Science*, 13 (1967) B359-B377.

Woodworth, M., S. Shanahan, Identifying the Critical Sequence in a Resource Constrained Project, *Project Management* 6 (1988) 89-96.

Zweben, M., Davis, E., Daun, B., Drascher, E., Deale, M., & Eskey, M., Learning to Improve Constraint-Based Scheduling, *Artificial Intelligence* 58 (1992) 271-296.